Thesis/
Reports
Pope,
R. P.

Alternative Models for Surface Resources

TECHNICAL USER'S MANUAL - GOAL: MULTIPLE OBJECTIVE PROGRAMMING

Report No. 3

SURFACE ENVIRONMENT AND MINING
Forest Service
U.S. Department of Agriculture

Department of Range Science
College of Forestry and Natural Resources
Colorado State University

Randal P. Pope

and

Kenneth E. Bottoms

February, 1976

Jointly supported by
Regional Analysis and Management of Environmental Systems
College of Forestry and Natural Resources
Colorado State University
Fort Collins, Colorado
N.S.F. Grant No. ESR72-03396-A02
and
SEAM Project
Billings, Montana
Grant No. 16-479-CT

ABSTRACT

Goal programming, as a general mathematical optimization technique and as incorporated in a specific computer algorithm, is discussed in this report. Use of this document is intended for those who desire to make changes in the computer program or who encounter dysfunctions which require correction. In addressing the technical considerations of this computer program, the areas of revised simplex-product form of the inverse, non-zero storage techniques, implicit objective function creation, and input/output specifications are discussed. The General User's Manual will be accompanied by a listing of the program and examples of problem execution.

The user not interested in modifying the computer program is referred to the General User's Manual for specifics as to the use of the goal programming algorithm. This manual can be obtained by writing to Regional Systems Program, 325 Aylesworth Hall, Range Science Department, CSU, Fort Collins, Colo. 80523.

TABLE OF CONTENTS

PURPOSE

This procedure finds the minimum of a multivariable, multiobjective linear system, subject to linear constraints:

Minimize $z_j = c_{j1}x_1 + c_{j2}x_2 + \ldots\ldots + c_{jN}x_N$

$j = 1,2, \ldots\ldots P$

Subject to $a_{i1}x_1 + a_{i2}x_2 + \ldots\ldots + a_{iN}x_N \leq, =, \geq$, or

either $\leq$ or $\geq$ b_i

$i = 1,2,\ldots, M$

and

$x_1, x_2, \ldots, x_N \geq 0$

where the a_{ik} terms are the technological coefficients, the b_i terms are the constraint right-hand-side values, the c_{jk} terms are the objective function coefficients, and the x_k terms are the decision variables; M is the number of constraint rows, P is the number of objective functions, and N is the number of decision variables.

METHOD

This method accomplishes the same results as the original simplex method applied to goal programming (Lee, 1972), but in a manner which is more efficient for implementation on a digital computer. In general, it computes only the current value of the rows or columns as required at each stage, and stores this information in a more concise form. Zero coefficients are neither input nor stored during computation.

Since revised simplex makes explicit use of matrix algebra, it is convenient to describe the problem in matrix notation:

Minimize $\underline{z} = C\ \underline{x}$

Subject to $A\ \underline{x} = \underline{b}$ and $\underline{x} \geq \underline{0}$ (zero vector)

where $\underline{x}$, $\underline{b}$, and $\underline{0}$ are column vectors,

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \qquad \underline{b} = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_M \end{bmatrix} \qquad \underline{0} = \begin{bmatrix} 0_1 \\ 0_2 \\ \cdot \\ \cdot \\ \cdot \\ 0_N \end{bmatrix} \qquad \underline{z} = \begin{bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_P \end{bmatrix}$$

C is a P x N matrix,

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ c_{P1} & c_{P2} & \cdots & c_{PN} \end{bmatrix}$$

and A is an M x N matrix,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdot & & \cdot & \\ \cdot & & \cdot & \\ \cdot & & \cdot & \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix}$$

A column vector, $\hat{\underline{x}}$, is introduced for the computational variables required for initial solution, with

$$\hat{\underline{x}} = \begin{bmatrix} x_{N+1} \\ x_{N+2} \\ \cdot \\ \cdot \\ \cdot \\ x_{N+S} \end{bmatrix}$$

bringing the total number of variables $(x_1, x_2, \cdots x_N, x_{N+1}, \cdots x_{N+S})$ to N+S. An M x S matrix $\hat{A}$ is formed to represent the technological coefficients relating to the computational variables $(x_{N+1}, x_{N+2}, \cdots x_{N+S})$ with

$$\hat{A} = \begin{bmatrix} a_{1,\ N+1} & a_{1,\ N+2} & \cdots & a_{1,\ N+S} \\ a_{2,\ N+1} & a_{2,\ N+2} & \cdots & a_{2,\ N+S} \\ \vdots & \vdots & & \vdots \\ a_{M,\ N+1} & a_{M,\ N+2} & \cdots & a_{M,\ N+S} \end{bmatrix} .$$

Similarly, a P x S order matrix $\hat{C}$ is formed containing the objective function coefficients relating to the computational variables, with

$$\hat{C} = \begin{bmatrix} c_{1,\ N+1} & c_{1,\ N+2} & \cdots & c_{1,\ N+S} \\ c_{2,\ N+1} & c_{2,\ N+2} & \cdots & c_{2,\ N+S} \\ \vdots & \vdots & & \vdots \\ c_{P,\ N+1} & c_{P,\ N+2} & \cdots & c_{P,\ N+S} \end{bmatrix}$$

The standard form:

$$\text{Minimize: } \underline{z} = C\ \underline{x}$$
$$\text{Subject to: } A\ \underline{x} = \underline{b}$$

can be shown as a single matrix equation

$$Y\ \underline{x} = \underline{w}$$

where Y is the augmented matrix of order (P+M) x (N+S)

$$Y = \left[\begin{array}{c|c} C & \hat{C} \\ \hline A & \hat{A} \end{array}\right]$$

$\underline{x}$ is the augmented vector of size N+S

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_N \\ x_{N+1} \\ x_{N+2} \\ \cdot \\ \cdot \\ \cdot \\ x_{N+S} \end{bmatrix}$$

and $\underline{w}$ is the vector of size P+M which is the concatenation of the vectors $\underline{z}$ and $\underline{b}$; i.e.,

$$\underline{w} = \left[\begin{array}{c} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_P \\ \hline b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_M \end{array}\right]$$

A solution to the equation

$$Y \, \underline{x} = \underline{w}$$

can also be expressed as

$$Y^{-1} \, Y \, \underline{x} = Y^{-1} \, \underline{w}$$

where Y^{-1}, rather than denoting the actual inverse of the matrix Y (which is not well-defined, since Y is non-square), denotes the inverse of an (M+P) x (M+P) submatrix of Y representing the variables in solution. The importance of this arrangement is that any value in the current set of equations can be calculated directly by performing the appropriate matrix operations.

The algorithm is as follows:

(1) The original matrix is generated representing the constraints and the objective functions. The artificial, slack, surplus, negative deviational and positive deviational variables are computed and adjoined to the matrix.

(2) The current value of objective function row j is calculated. The jth row will be operated on until z_j is at its minimum, given that the minimum has been reached at all preceding levels of j. The current value of the jth row is found by

$$(Y^{-1})^j \; Y = \overline{Y}^j$$

where Y is multiplied by the jth row of the inverse matrix Y^{-1}, yielding the current value of a single row in the Y matrix (super bar indicating current value, and superscript j indicating the jth row of the matrix). From this current row the largest positive value is chosen, indicating the column to be brought into solution. This column is denoted by subscript t. In the case where no positive value is found, j is increased by 1 and if j equals P, an optimal solution has been reached.

(3) The current value of column t ($\overline{Y}_t$) is determined by

$$\overline{Y}_t = Y^{-1} \; Y_t$$

and the variable leaving solution is found by locating the minimum $\overline{w}_i/\overline{y}_{it}$ which is positive or zero. This row is denoted by r.

(4) A pivot vector is formed to update Y^{-1} by

$$\begin{bmatrix} -\bar{c}_{1t}/\bar{a}_{rt} \\ -\bar{c}_{2t}/\bar{a}_{rt} \\ \cdot \\ \cdot \\ \cdot \\ -\bar{c}_{Pt}/\bar{a}_{rt} \\ -\bar{a}_{1t}/\bar{a}_{rt} \\ -\bar{a}_{2t}/\bar{a}_{rt} \\ \cdot \\ \cdot \\ 1/\bar{a}_{rt} \\ \cdot \\ \cdot \\ \cdot \\ -\bar{a}_{Mt}/\bar{a}_{rt} \end{bmatrix}$$

where $1/\bar{a}_{rt}$ indicates the pivot element, and occurs as the rth entry of the pivot vector.

(5) Update the right-hand-side values by

$$Y^{-1}\,\underline{w} = \underline{\bar{w}}$$

(6) Return to step 2.

The above procedure describes the conceptual process of the algorithm, although in fact Y^{-1} is stored in product form and the objective function rows are not explicit in the matrix but are generated as needed.

PROGRAM DESCRIPTION

Usage - Main Program GOAL

The entire GOAL program is written to comply with the FORTRAN demands of the American National Standards Institute (ANSI) in order that the program be implementable on the widest variety of computer systems. To further enhance GOAL's transportability, variable names (READU and WRITEU) have been assigned to the input and output unit numbers, respectively, so that the program may be easily used with varying input and output devices, either in batch mode or interactively, by simply changing a single DATA statement in the BLOCK DATA subprogram. For example, to use the program in ordinary batch mode on a system where the input unit number is 5 and the output unit number is 6, the following card must be supplied in the BLOCK DATA subprogram:

```
DATA READU/5/, WRITEU/6/
```

The unit numbers (5 and 6 in the example) will vary depending on the system and the particular choice of input and output devices being used, and must be supplied by the user in accordance with the requirements of his system.

The program consists of a main program and 15 subroutines. The configuration of the program, in terms of special duty subroutines, is based mainly on the intention of using a command vocabulary set to control the specific operation of the program during execution. The general flow chart for the main program is shown in Fig. 1.

The main program, upon initial execution, immediately passes execution to subroutine INPT, in which the commands to the program are read and interpreted. Execution of the main program occurs only after the command STRT (a Type 1 Control Card) has been read by INPT. When execution is returned to the main program, the GO TO statement following the CALL INPT sends execution to the READ statement. This statement reads the parameters of a particular problem which is to be solved, where NA is the number of rows, NB is the number of columns, MB is the number of priority levels, NNN is the number of non-zero matrix entries (the number of MTRX data cards), NNG is the number of G type constraints, NNL is the number of L type constraints, NNB is the number of B type constraints, NAA is a non-zero positive integer when intermediate execution results are desired, and NBB is a non-zero positive integer when input data echo is desired.

The parameters NA, NB, NNN, NNG, NNL, and NNB are used to calculate the values of N1 through N25. The value of these variables will be used to identify the first word of a section into which the blank COMMON block with array D is divided. Each of these subdivisions of D will be passed as formal parameters into the other subroutines. The purpose of this configuration is to allow for changes in core requirements, given the size of the problem to be solved. The only change needed is the dimension of array D so that the dimension of D is greater

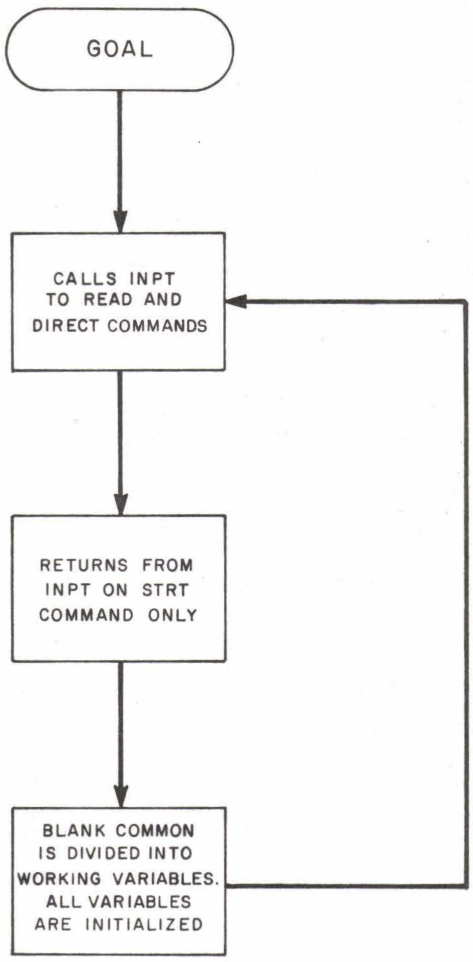
GOAL
CALLS INPT TO READ AND DIRECT COMMANDS
RETURNS FROM INPT ON STRT COMMAND ONLY
BLANK COMMON IS DIVIDED INTO WORKING VARIABLES. ALL VARIABLES ARE INITIALIZED

Fig. 1. The general flow chart for the main program.

than N25. Some operating systems allow for a system request which will expand the amount of blank COMMON during program execution.

To summarize the use of the array D in blank COMMON:

KG	WW	JG	IZIP	MG	
D(1)	D(N1)	D(N2)	D(N3)	D(N4)	D(N5)

where the variables N1 through N24 designate the first word in each subdivision of the array D, except that the first subdivision begins with 1. The last subdivision ends with N25.

The individual subdivisions of D will be known by the following variable names in all of the subroutines:

<u>Beginning Word</u>	<u>Ending Word</u>	<u>Formal Parameter Variable Name</u>
D(1)	D(N1-1)	KG
D(N1)	D(N2-1)	WW
D(N2)	D(N3-1)	JG
D(N3)	D(N4-1)	IZIP
D(N4)	D(N5-1)	MG
D(N5)	D(N6-1)	DY
D(N6)	D(N7-1)	A
D(N7)	D(N8-1)	IA
D(N8)	D(N9-1)	KL
D(N9)	D(N10-1)	KN
D(N10)	D(N11-1)	NR
D(N11)	D(N12-1)	B
D(N12)	D(N13-1)	JH
D(N13)	D(N14-1)	KB
D(N14)	D(N15-1)	X
D(N15)	D(N16-1)	Y
D(N16)	D(N17-1)	KNT
D(N17)	D(N18-1)	LABL
D(N18)	D(N19-1)	W

D(N19)	D(N20-1)	CAX
D(N20)	D(N21-1)	IKJ
D(N21)	D(N22-1)	E
D(N22)	D(N23-1)	IE
D(N23)	D(N24-1)	IP
D(N24)	D(N25)	LE

The nexւ instructions of the main program initialize at zero all variables in blank and labeled COMMON, after which those variables not desired at a zero level are initialized at their proper values.

Subroutines Required

SUBROUTINE ALTR (K, KK, COMMON PARAMETERS) processes any changes from the original input data. This includes changes in right-hand-side values, technological coefficients, priority levels, differential weights and the sense of the constraint inequalities. Parametric runs on technological coefficients and right-hand-side values are performed. The flow chart for SUBROUTINE ALTR is shown in Fig. 2.

Unique formal parameters:

K - passes back to subroutine INPT the command encountered after subroutine ALTR has completed its purpose.

KK - designates the specific task being performed by ALTR. IF KK is:

1 - a change in a matrix entry is being made;

2 - a change in any of the data contained in a RHS card is being made;

3 - a parametric run on either a matrix entry or a RHS is requested;

4 - a parametric run on either a matrix entry or a RHS is in progress.

```
J3=0
KM(16)=0
KM(1)=0
```

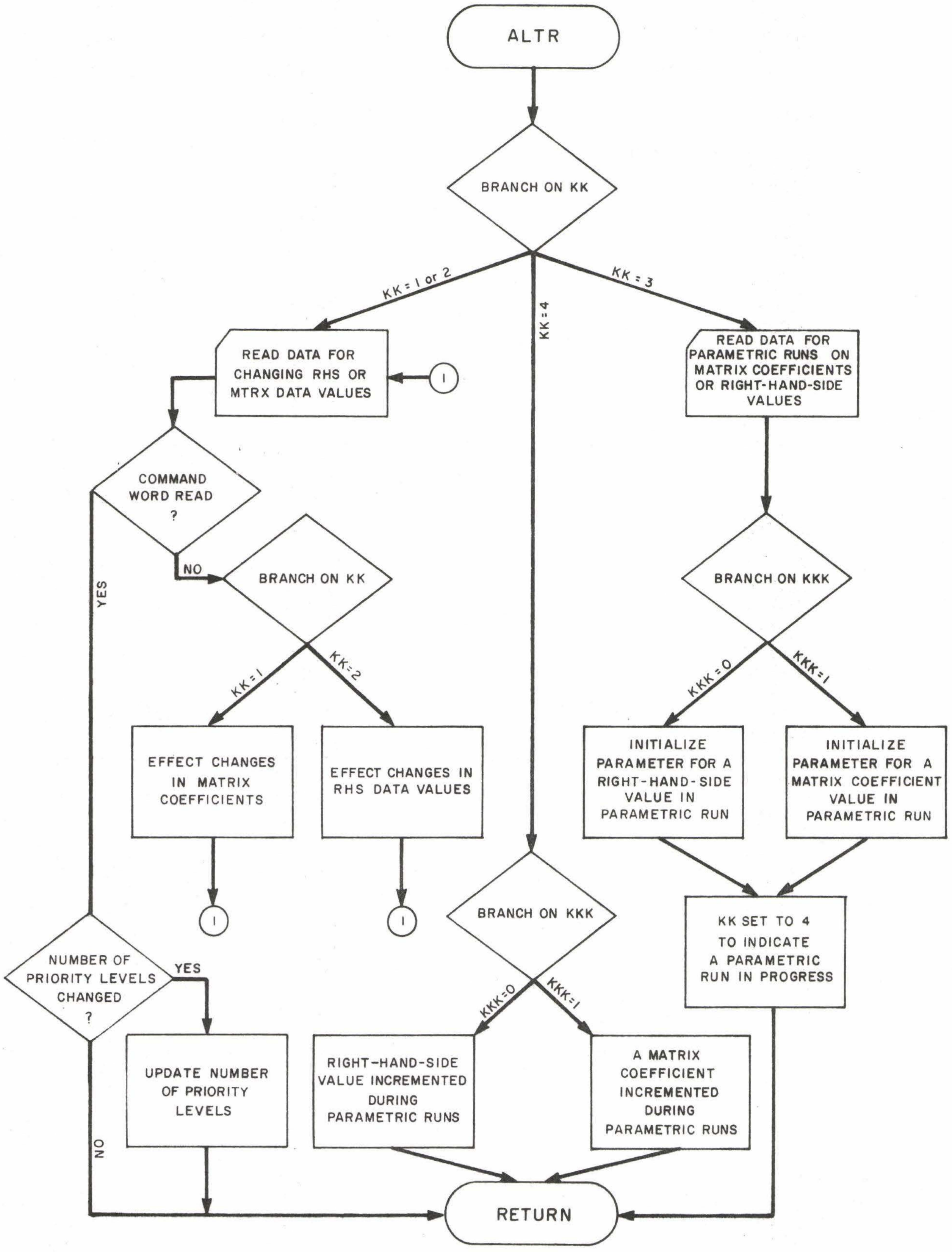
ALTR
BRANCH ON KK
KK = 1 or 2
KK = 4
KK = 3
READ DATA FOR CHANGING RHS OR MTRX DATA VALUES
1
READ DATA FOR PARAMETRIC RUNS ON MATRIX COEFFICIENTS OR RIGHT-HAND-SIDE VALUES
COMMAND WORD READ ?
NO
YES
BRANCH ON KK
BRANCH ON KKK
KK=1
KK=2
KKK=0
KKK=1
EFFECT CHANGES IN MATRIX COEFFICIENTS
EFFECT CHANGES IN RHS DATA VALUES
INITIALIZE PARAMETER FOR A RIGHT-HAND-SIDE VALUE IN PARAMETRIC RUN
INITIALIZE PARAMETER FOR A MATRIX COEFFICIENT VALUE IN PARAMETRIC RUN
1
1
BRANCH ON KKK
KK SET TO 4 TO INDICATE A PARAMETRIC RUN IN PROGRESS
NUMBER OF PRIORITY LEVELS CHANGED ?
YES
NO
KKK=0
KKK=1
UPDATE NUMBER OF PRIORITY LEVELS
RIGHT-HAND-SIDE VALUE INCREMENTED DURING PARAMETRIC RUNS
A MATRIX COEFFICIENT INCREMENTED DURING PARAMETRIC RUNS
RETURN

Fig. 2. Flow chart for SUBROUTINE ALTR.

J3 is a variable which is zero if no changes have been made on priority levels and 1 if changes have been made. KM(16), the priority level being operated on, and KM(1), the number of iterations involved in problem solution, are set to zero under the assumption that after the changes have been made in the data set, a solution will be sought for the problem.

```
      GO TO (101,101,118,122),KK
```

KK controls the use of the SUBROUTINE ALTR. If KK is 1 or 2, execution goes to the READ statement below where processing begins for change in either RHS or MTRX data entries. If KK is 3, then processing goes to the second READ statement to process data for a parametric run. If KK is 4, a parametric run is in progress and execution is sent to the instructions which increment the parameter.

```
  101 READ(READU,134)K,K1,K2,AA,IT,IG,PW,LG,QW
  134 FORMAT(3A4,F12.0,A1,2(I5,F12.0))
      IF(K.NE.KM(25)) GO TO 128
      GO TO (102,104),KK
```

Here the data is read which will effect changes in either the RHS or MTRX data entries.

Cols. 1-4	K	- command word.
Cols. 5-8	K1	- column name or symbol.
Cols. 9-12	K2	- row name or symbol.
Cols. 13-24	AA	- value of the RHS of a constraint if columns 5-8 are blank, or the value of a matrix entry if fields 5-8 and 9-12 are used.
Col. 25	IT	- type of equality (E, G, L, B).
Cols. 26-30	IG	- priority level of negative deviation assigned to the row in columns 9-12.
Cols. 31-42	PW	- differential weight assigned to priority level in columns 26-31.
Cols. 43-47	LG	- priority level of positive deviation assigned to the row in columns 9-12.
Cols. 48-59	QW	- differential weight assigned to priority level in columns 43-47.

It should be noted that this READ statement handles data changes for both RHS and MTRX data entries. If a change is for a MTRX entry then only columns 5-24 are used. If a change is for a RHS entry, then columns 9-59 are relevant.

After the READ is effected, a check is made on K (contents of columns 1-4) to insure that the data is not a command word. If a command word has been read, execution goes to necessary checks which must be made for exiting from ALTR.

If no command word has been read, the execution is allowed to continue with a GO TO statement which directs processing for the data which has been read, depending on whether a RHS or MTRX entry is to be changed.

```
102 KJ=N-KM(14)-KM(26)
    DO 103 J=1,KJ
    JJ=J
    IF(K1.EQ.KN(J)) GO TO 104
103 CONTINUE
    GO TO 125
```

KJ is the number of columns, excluding any computational columns.

The DO loop searches for column names, given in either the first READ statement of this routine, or in the READ statement associated with parametric runs. JJ will contain the number of the column associated with the column name or symbol. When the column name is found, K1 is equal to KN(J), then execution is passed to the row name processor.

If the DO loop is executed to its upper bound and no match is found for a column name, then the GO TO following the DO loop CONTINUE statement sends execution to a diagnostic indicating an error in the input data.

```
104 DO 105 I=1,M
    II=I
    IF(K2.EQ.NR(I)) GO TO 106
105 CONTINUE
    GO TO 125
```

This DO loop is used in all data change processing or parametric runs. It searches the names of all rows, NR(I), to find the one which agrees with K2. II will contain the number of that row. When the row name is found, K2 is equal to NR(I), then execution goes to a GO TO statement which will direct further execution.

If no row name match is found after going through the total number of rows (M), the GO TO following the DO loop CONTINUE statement sends execution to a diagnostic indicating an error in the row name input data.

```
106 GO TO (107,111,119),KK
```

After a match-up in row names has been made, execution goes to a loop to find the number of a matrix entry if a MTRX entry change is being made (KK is 1). Execution goes to instructions to process RHS data changes if KK is 2; or execution is sent to an IF statement to determine if a parametric run is being done on a matrix entry or a right-hand-side value (KK is 3).

```
107 LUMP1=KL(JJ)
    LUMP2=KL(JJ+1)-1
    DO 108 I=LUMP1,LUMP2
    III=I
    IF(IA(I).EQ.II) GO TO 109
108 CONTINUE
```

This block of instructions identifies the number of a matrix entry (III). Either a change in MTRX data or a parametric run on a matrix entry will use these instructions. LUMP1 is the number of the first matrix entry in column JJ. LUMP2 is the number of the last matrix entry in column JJ. This is computed by subtracting 1 from the number of the first matrix entry in the following column (JJ+1). IA(I) is the row number associated with matrix entry A(I). When the desired row number (II) corresponding to a matrix entry row is found, execution is sent to a GO TO statement which will determine further execution.

```
109 GO TO (110,110,120),KK
```

When KK is 2, execution is sent to the same label as if KK is 1. This occurs because KK being 2 is not applicable in this situation, yet the 2 position is required in the computed GO TO. If a MTRX entry is to be changed (KK is 1), then execution is sent to the instruction which will make the change. If a parametric run is being initiated (KK is 3), then execution is sent to where the initial value of the matrix entry is set.

```
110 A(III)=AA
    GO TO 101
```

When a MTRX data entry is to be changed, this is where the actual change is made. After the change is effected, the GO TO sends execution back to the first READ statement to process any other changes or read a command word.

```
111 B(II)=AA
    DO 112 I=1,4
    IK=I
    IF(INAM(I).EQ.IT) GO TO 113
112 CONTINUE
113 IZIP(II)=IK
    KM(34)=1
    IF(IG.GT.0) GO TO 114
    KG(II)=999
    GO TO 115
114 KG(II)=IG
115 J3=1
    WW(II)=PW
    IF(LG.GT.0) GO TO 116
    MG(II)=999
    GO TO 117
116 MG(II)=LG
117 DY(II)=QW
    GO TO 101
```

This block of instructions processes the changes in RHS data entries. The DO loop in this block changes the alphanumeric symbols E, G, L, B (the inequality types of constraints) to the numerics 1, 2, 3, 4, respectively.

J3 is set to 1 to indicate that a change has been made in a priority level. When processing is completed in ALTR, a check is made to see if J3 is equal to 1. If so, a check is made to see if the number of priority levels increased or decreased.

KM(34) is set to 1 if any changes are made which will affect the configuration of computational variables. In SUBROUTINE INPT, action is taken if KM(34) is equal to 1.

```
118 READ(READU,135)K,K1,K2,T1,T2,T3
135 FORMAT(3A4,3F12.0)
```

This READ statement is used when a parametric run will be processed.

Cols.	1-4	K	- blank (this should never contain even a command word).
Cols.	5-8	K1	- name of column if parametric run is on a matrix entry.
Cols.	9-12	K2	- name of the row corresponding to K1 if a matrix entry is being run, or the name of a row if a parametric run is on a right-hand-side value.

Cols. 13-24	T1 - the initial value of the parameter.
Cols. 25-36	T2 - the terminal value of the parameter.
Cols. 37-48	T3 - the increment which will be added to the parameter at each solution.

```
      IF(K1.EQ.KM(25)) GO TO 104
      KKK=1
```

The IF statement checks to see if the variable which contains the column name is blank. If so, it is assumed that a parametric run is being done on a right-hand-side value, and execution is sent to the DO loop which will identify the row number (II).

If the K1 field is not blank, it is assumed that the parametric run will be on a matrix entry. KKK is set to 1, which indicates that processing will be on a matrix entry parametric run.

```
      GO TO 102
```

This command sends execution to find the column number (JJ), the row number (II), and the matrix entry number (III) which pertain to the particular parametric run.

```
  119 IF (KKK.EQ.1) GO TO 107
```

After II has been determined, execution is sent to this instruction to check whether or not a matrix entry is to be processed. If so, execution is sent to find III, the number of the matrix entry.

```
      B(II)=T1
      GO TO 121
```

The right-hand-side (B(II)) is set to the initial value (T1) desired in the parametric run. The GO TO sends execution to intermediate parametric run processing.

```
  120 A(III)=T1
  121 KK=4
      GO TO 126
```

The initial value (T1) of matrix entry A(III) is set. KK is set to 4, indicating that a parametric run is in progress. The GO TO sends execution to appropriate output statements, then to RETURN.

```
122 IF(KKK.EQ.1) GO TO 124
    KM(35)=4
    B(II)=B(II)+T3
    IF(B(II).LE.T2) GO TO 126
123 KK=5
```

The first IF statement in this block of instructions is where execution comes after the initial values have been set in a parametric run, the solution at that parameter value has been found, and execution has been returned to ALTR for incrementing the parameter value.

If KKK is 1, the execution is sent to where a matrix entry will be incremented; if KKK is 0, then KM(35) is set to 4 so that the output of the solution at that parameter value will not have a complete heading.

The right-hand-side, B(II), is then incremented by the amount T3. A check is made to see if the increment forced the right-hand-side over the desired upper bound, T2. If not, the execution is sent to print out the subheading for the parametric run solution. If the check fails, KK is set to 5, which indicates that the parametric run should be terminated when execution is returned to INPT.

```
    KM(35)=1
    GO TO 133
```

KM(35) is set to 1 so that a full printout will be received after the parametric run is terminated. The GO TO sends execution to the RETURN statement.

```
124 A(III)=A(III)+T3
    KM(35)=4
    IF(A(III).LE.T2) GO TO 126
    GO TO 123
```

A parametric run on a matrix entry is in progress. The value of the matrix entry A(III) is incremented by T3. KM(35) is set to 4 to suppress the main output heading during parametric runs. A check is made to insure that the new value of the matrix entry does not exceed the upper bound, T2. If the check holds, then instructions for printing the subheading in the parametric solution output is executed. The GO TO is executed if the above check does not hold, sending execution to terminating instructions.

```
125 WRITE(WRITEU,136) K1,K2
136 FORMAT(1H0,7HEITHER ,A4,4H OR ,A4,33H ARE ILLEGAL COLUMN OR ROW
   1 NAMES.)
    STOP
```

This block is the diagnostic output when the column or row names read in cannot be matched with existing names.

```
126 IF(KKK.EQ.1) GO TO 127
```

If KKK is 1 (a parametric run is being made on a matrix entry), execution is sent to the proper subheading WRITE statements. If KKK is 0, the proper WRITE statements for right-hand-side subheading are shown below.

```
    WRITE(WRITEU,139)
    WRITE(WRITEU,137) NR(II)
137 FORMAT(1H ,2H$$,17X,29HRIGHT-HAND-SIDE VALUE OF ROW ,A4,17X,
   12H$$)
    WRITE(WRITEU,140) B(II)
    GO TO 133
```

WRITE statements for parametric run subheading on right-hand-side value.

```
127 WRITE(WRITEU,139)
    WRITE(WRITEU,138) KN(JJ),NR(II)
138 FORMAT(1H ,2H$$,14X,20HMATRIX ENTRY-COLUMN ,A4,10H WITH ROW ,
   1A4,15X,2H$$)
    WRITE(WRITEU,140) A(III)
    GO TO 133
```

WRITE statements for parametric run subheading on matrix entry values.

```
139 FORMAT(1H0,28(1H$),14HPARAMETRIC RUN,29(1H$)/1H ,2H$$,31X,
   12HON,34X,2H$$)
140 FORMAT(1H ,2H$$,27X,13HAT A VALUE OF,27X,2H$$/1H ,2H$$,26X,
   1F12.3,29X,2H$$/1H ,71(1H$))
```

FORMATS shared by both right-hand-side and matrix entry parametric run subheadings.

```
128 IF(J3.EQ.0) GO TO 133
    IBIG=0
    DO 132 I=1,M
    IF(KG(I).EQ.999) GO TO 129
    IF(KG(I).GT.IBIG) GO TO 130
129 IF(MG(I).EQ.999) GO TO 132
    IF(MG(I).GT.IBIG) GO TO 131
    GO TO 132
130 IBIG=KG(I)
    GO TO 129
131 IBIG=MG(I)
132 CONTINUE
    MB=IBIG
133 RETURN
```

The first IF statement checks J3 to see if it is zero (indicating no priority level changes were made). If the check holds, execution is sent to RETURN. If the test does not hold, all priority levels are checked and the value of the highest (numerically) to date is stored in IBIG. After all priority levels are checked, MB (the number of priority levels) is set to IBIG. Then execution is sent to RETURN.

SUBROUTINE ART (COMMON PARAMETERS) initializes arrays used in controlling columns in solution, and other solution-involved variables. The principal task of this subroutine is to create all computational variable columns (slacks, surpluses, negative deviations, and positive deviations) and establish which columns are in solution in which rows. Rows which have artificial variables in solution are also identified. The flow chart for SUBROUTINE ART is shown in Fig. 3.

```
    KM(36)=1
    LE(1)=0
```

KM(36) is set to 1 indicating that the computational columns have been created. Setting LE(1) to zero indicates that the problem has yet to be solved.

```
    K2=N+2*M+1
    DO 101 I=1,K2
101 IKJ(I)=0
```

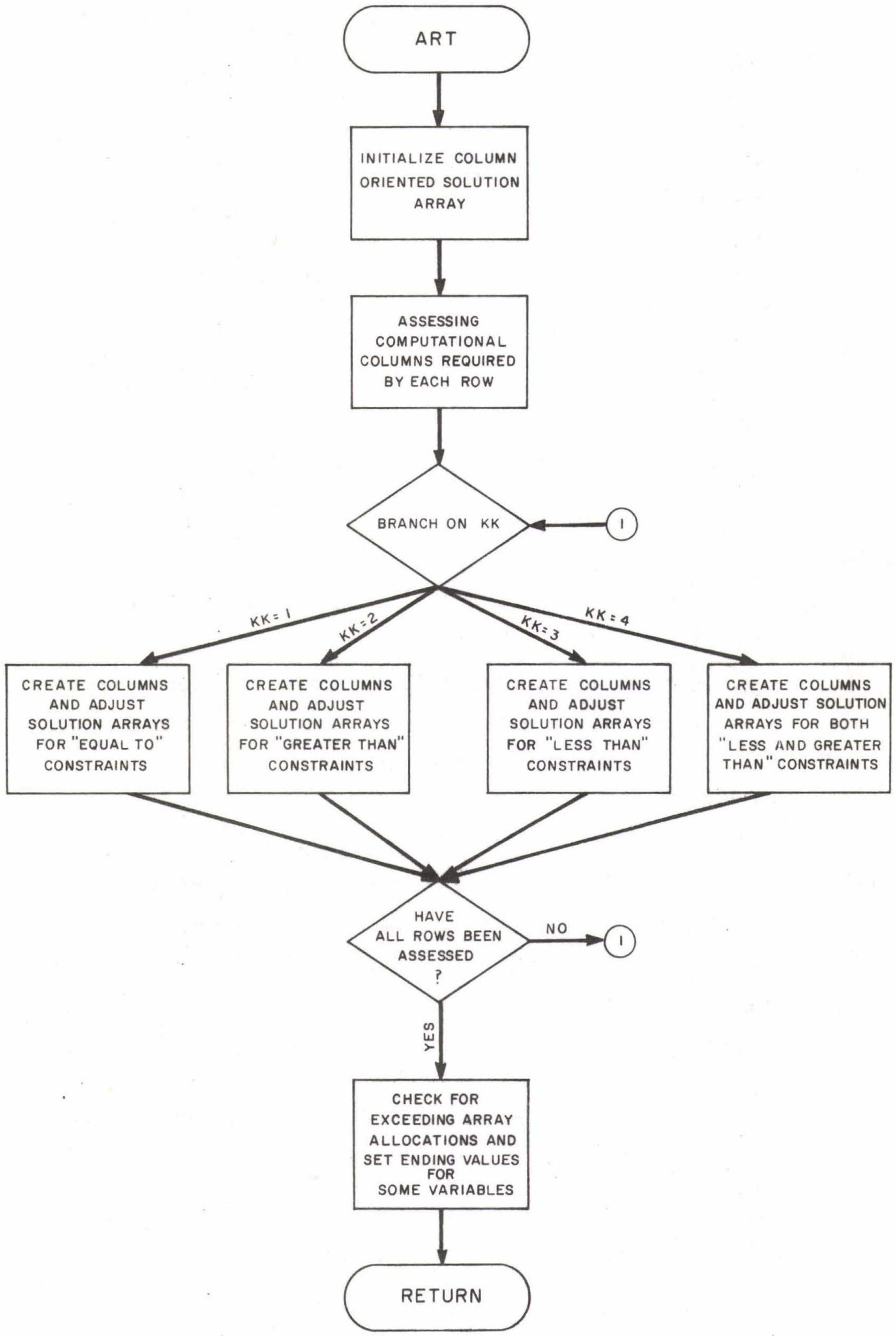
ART
INITIALIZE COLUMN ORIENTED SOLUTION ARRAY
ASSESSING COMPUTATIONAL COLUMNS REQUIRED BY EACH ROW
BRANCH ON KK
I
KK = 1
KK = 2
KK = 3
KK = 4
CREATE COLUMNS AND ADJUST SOLUTION ARRAYS FOR "EQUAL TO" CONSTRAINTS
CREATE COLUMNS AND ADJUST SOLUTION ARRAYS FOR "GREATER THAN" CONSTRAINTS
CREATE COLUMNS AND ADJUST SOLUTION ARRAYS FOR "LESS THAN" CONSTRAINTS
CREATE COLUMNS AND ADJUST SOLUTION ARRAYS FOR BOTH "LESS AND GREATER THAN" CONSTRAINTS
HAVE ALL ROWS BEEN ASSESSED ?
NO
I
YES
CHECK FOR EXCEEDING ARRAY ALLOCATIONS AND SET ENDING VALUES FOR SOME VARIABLES
RETURN

Fig. 3. Flow chart for SUBROUTINE ART.

K2 is the number of words allocated to array IKJ in the main program (N21-N20). This DO loop sets IKJ(I) (the row in which column I is in solution) to zero.

```
          DO 119 I=1,M
          KKK=0
          KK=IZIP(I)
          GO TO (102,104,104,104),KK
```

This DO loop evaluates each row from 1 to M. KKK is equal to zero when the negative deviation for a B type constraint is being created, and KKK is 1 when the positive deviation for a B type constraint is being created.

KK indicates the type of inequality required by row I.

1 (equivalent to E) - equal to constraint.

2 (equivalent to G) - greater than constraint.

3 (equivalent to L) - less than constraint.

4 (equivalent to B) - both type constraint.

The GO TO statement directs execution to the proper instruction, given the inequality type.

```
      102 IF(KG(I).EQ.999) GO TO 113
          GO TO (104,119,118,118),KK
```

The IF statement checks negative deviational variable priority level for the value of 999. This indicates that the negative deviation for row I has no priority level.

```
      103 KKK=1
      104 N=N+1
          MA=MA+1
          KL(N)=MA
          GO TO (105,114,102,111),KK
```

KKK is set to 1 only after the negative deviation for a B type constraint has been created.

The next 3 instructions are executed whenever a column is created. N is the number of the last column, MA is the number of the last matrix entry, and KL(N) is the number of the first matrix entry in column N.

```
105 KN(N)=EQLD
106 JG(N)=KG(I)
    A(MA)=1.0
    GO TO 108
```

If row I has constraint type 1 and there is a priority level associated with it (KG(I) not equal to 999), then a column is assigned to that variable and the column is designated an EQLD variable. The column is given the value associated with that variable (JG(N)=KG(I)), and the matrix entry A(MA) is given the value of 1.

This small block is used when the row requires:

- an E associated with a priority level.
- an L regardless of priority level.
- or a B regardless of priority level.

```
107 KM(14)=KM(14)+1
    JG(N)=MG(I)
    A(MA)=-1.0
```

This block of instructions is used any time that the column created is for a row requiring a G inequality or a B. JG(N) is the priority level assigned to the variable in column N. The matrix entry A(MA) for any surplus type variable (actual surpluses or positive deviational variables) is given the value of -1. KM(14) keeps track of the number of surplus type variables created.

```
108 KB(N)=I
    IA(MA)=I
    GO TO (109,110,109,112),KK
```

Each column is created because of a row requirement. KB(N) contains the number of the row which required that column N be created. IA(MA) is the number of the row in which matrix entry MA falls.

```
109 IKJ(N)=I
    JH(I)=N
    KM(26)=KM(26)+1
    GO TO (119,119,119,103),KK
```

This block is used only when slack or negative deviational variable columns are created (when row requirement for inequalities are L or B). Every column created for a slack or negative deviation is put into solution in the row (I) which required the variable. JH(I), therefore, indicates that column N is in solution in row I. Conversely, IKJ(N) indicates that column N is presently in solution in row I. IKJ is column oriented while JH is row oriented. KM(26) is incremented as slack type variables are added (these are actual slacks and negative deviations).

```
110 JH(I)=0
    KM(8)=KM(8)+1
    GO TO 119
```

This block is used when a row requires an E (and has no priority level) or a G. JH(I) equaling zero shows that no column is in solution in the row, therefore a zero JH(I) implies an artificial is in solution. KM(8) keeps track of the number of artificials created.

```
111 IF(KKK.EQ.0) GO TO 118
    GO TO 115
112 IF(KKK.EQ.0) GO TO 109
    GO TO 119
```

When KK is 4 (a "B" type constraint is required), these IF statements help control the processing of both a negative and a positive deviation column. KKK is 1 when positive deviations are being processed and 0 when negative deviations are being processed.

```
113 GO TO (110,119,117),KK
```

When negative deviations have a priority level of 999 (indicates no priority level), processing is sent to the above GO TO statement to direct further processing. Since KK being 2 does not apply to negative deviations, the statement number in the second position of the computed GO TO is just a place holder.

```
114 IF(MG(I).EQ.999) GO TO 116
115 KN(N)=POSD
    GO TO 107
116 KN(N)=SRPL
    GO TO 107
```

Positive deviational variables (POSD) and surplus variables (SRPL) are assigned as column names. When MG(I) is 999, it indicates that a positive deviation has no priority level.

```
117 KN(N)=SLCK
    GO TO 106
118 KN(N)=NEGD
    GO TO 106
119 CONTINUE
```

Any column N which is created by a request for an L type inequality without a priority level is designated SLCK for slack. Any request for an L type inequality with a goal level or a B type inequality (with or without a priority level on the negative deviation) is designated with NEGD for its column name to indicate a negative deviation.

The CONTINUE statement ends the DO loop which has stepped through each row number.

```
IF(MA.GE.KM(2)) GO TO 120
IF(N.GT.KM(12)) GO TO 121
```

The first instruction checks to see if the number of words allocated in the main routine for non-zero matrix entries has been exceeded. The second instruction does the same check for the number of columns.

```
KL(N+1)=MA+1
GO TO 122
```

During execution, reference to matrix entries within a column is made by determining the first matrix entry in a column, and the last matrix entry in the same column by subtracting 1 from the following column's number of its first matrix entry. Therefore, in order to find the number of matrix entries in the last column, it is necessary to create a fictional first matrix entry of the imaginary last column, N+1. KL(N+1) = MA + 1 accomplishes this task.

```
120 WRITE(WRITEU,125)
125 FORMAT(1H0,51HMAXIMUM NUMBER OF MATRIX ENTRIES HAS BEEN EXCEEDED.
   STOP 13
121 WRITE(WRITEU,126)
126 FORMAT(1H0,44HMAXIMUM NUMBER OF COLUMNS HAS BEEN EXCEEDED.)
    STOP 15
```

These are self-explanatory diagnostic messages.

```
122 IF(KM(37).LE.0) GO TO 124
    DO 123 I=1,N
    MTA=KL(I)
    MTB=KL(I+1)-1
    WRITE(WRITEU,127) KN(I),I
127 FORMAT(1H0,19HCONTENTS OF COLUMN ,A4,8H-NUMBER ,I5)
    WRITE(WRITEU,128) (A(J),J=MTA,MTB)
128 FORMAT(1H ,7(1X,F10.2))
    WRITE(WRITEU,129) (IA(K),K=MTA,MTB)
129 FORMAT(1H ,7(1X,I10))
123 CONTINUE
124 RETURN
```

KM(37) is an intermediate output control variable. In this case, if KM(37) is non-zero, then the complete matrix with its newly created columns is printed out. MTA is the number of the first matrix entry in each column, and MTB is the last.

SUBROUTINE BRO (II, IREJ, COMMON PARAMETERS) establishes the pricing vector at each iteration, multiplies the pricing vector by the eta file with a backward pass, multiplies each non-basic column of the matrix by the completed pricing vector, stores the resulting values by columns in array CAX, selects the largest positive value of CAX, and returns the column number of the highest value to SUBROUTINE PRI through formal parameter (argument) II (this is the pivot column). Also, if a column which has formerly been chosen is later rejected, BRO will be called by SUBROUTINE CNTRL and IREJ will be equal to 1. In this instance the CAX value associated with the pivot column will be set to zero, and the array CAX will again be searched for its highest positive value and a new pivot column will be established. CAX is technically the objective function of the current priority level, KM(16). The flow chart for SUBROUTINE BRO is shown in Fig. 4.

Unique formal parameters:

II - contains the number of the column selected as the pivot column, and returns that value either to CNTRL or PRI.

IREJ - when equal to 1 indicates that the column previously selected has not been matched with a pivot row. In this case BRO will

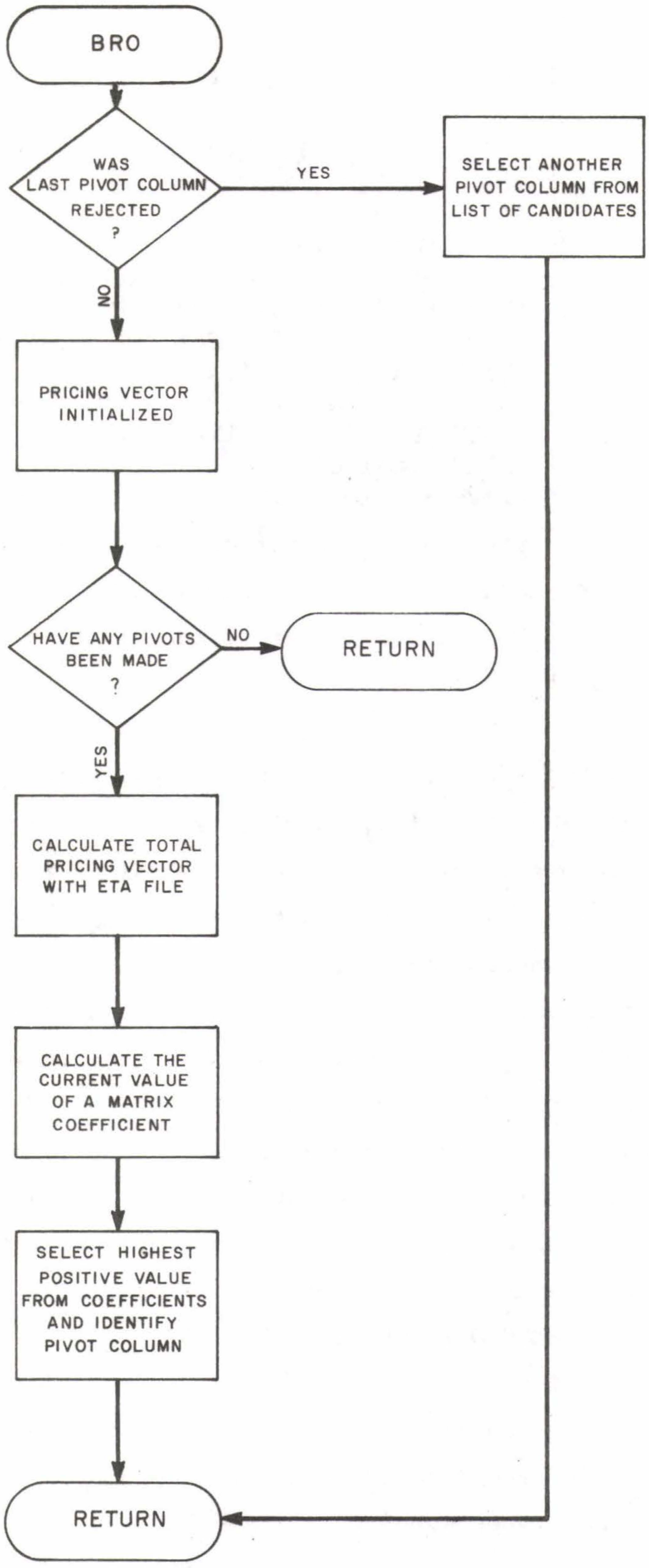
BRO
WAS LAST PIVOT COLUMN REJECTED ?
YES
SELECT ANOTHER PIVOT COLUMN FROM LIST OF CANDIDATES
NO
PRICING VECTOR INITIALIZED
HAVE ANY PIVOTS BEEN MADE ?
NO
RETURN
YES
CALCULATE TOTAL PRICING VECTOR WITH ETA FILE
CALCULATE THE CURRENT VALUE OF A MATRIX COEFFICIENT
SELECT HIGHEST POSITIVE VALUE FROM COEFFICIENTS AND IDENTIFY PIVOT COLUMN
RETURN

Fig. 4. Flow chart for SUBROUTINE BRO.

select the next highest positive value of CAX and return the corresponding column number back to CNTRL through parameter II.

```
      IF(IREJ.EQ.1) GO TO 122
```

If IREJ is 1, execution is directed to instructions which will set the CAX value of the former pivot column to zero, and the highest positive value of CAX is again selected.

```
      III=0
      KK=KM(16)
```

III is 1 if rows are found which have columns of the priority level being operated on in their solution.

```
      DO 101 I=1,M
  101 W(I)=0.0
```

W(I), the pricing vector, is initialized at zero.

```
      IF(KM(30))106,102,106
```

There are two ways to set up the pricing vector. When an infeasibility exists (KM(30) is 1), one set of instructions is used to set up the vector, and when no infeasibilities exist (KM(30) is 0), another set of instructions is used. The above IF statement directs execution to the proper set of instructions.

```
  102 DO 105 I=1,M
      JJ=JH(I)
      IF(JG(JJ).EQ.KK) GO TO 103
      GO TO 105
  103 III=1
      JI=KB(JJ)
      IF(KN(JJ).EQ.POSD) GO TO 104
      W(I)=WW(JI)
      GO TO 105
  104 W(I)=DY(JI)
  105 CONTINUE
```

This DO loop sets up the pricing vector if no infeasibilities exist. JJ is the number of the column in solution in row I. The following IF statement checks the priority level (JG) of the column to see if it is the same as the priority level being operated on (KK). If not, the pricing vector (W) at row I remains at zero, and control is sent to the end of the loop. If the priority level of the column in solution in row I is equal to the priority level being operated on, III will be set to 1 indicating that deviations at priority level KK do need to be minimized. JI is the row number of the original row which required that column JJ be created when the initial solution was formed. Next, the type of deviation is determined with the next IF statement (POSD for positive deviation, or default for negative deviation). If a negative deviation is in solution, the pricing vector at row I is set to the value of the differential weight (WW) associated with the original row JI. If a positive deviation is in solution, the pricing vector at row I is set to the value of the differential weight (DY) associated with the original row JI.

```
    IF(III.EQ.1) GO TO 111
    II=0
    GO TO 124
```

If III is 1, then deviations from priority level KK have been found in solution and execution is continued. If III is zero, then the pivot column II is set to zero and control is sent back to PRI where the priority level being operated on is incremented.

```
106 DO 110 I=1,M
    IF(X(I))107,108,108
107 W(I)=-1.0
    GO TO 110
108 IF(JH(I)) 110,109,110
109 W(I)=1.0
110 CONTINUE
```

This DO loop sets up the pricing vector if infeasibilities exist. If a right-hand-side value is negative, the pricing vector at row I is set to -1. If the right-hand-side is positive, but an artificial variable is in solution in row I (JH(I)=0 indicates an artificial), then the pricing vector at row I is set to 1.

```
111 IF(ME)112,115,112
```

After the pricing vector has been set up, this IF statement determines if it needs to be multiplied by the eta file. If ME is zero it does not.

```
112 DO 114 K6=1,ME
    K1=ME+1-K6
    T(2)=0.0
    LU3=LE(K1)
    LU4=LE(K1+1)-1
    DO 113 K3=LU3,LU4
    IMP=IE(K3)
    T(2)=T(2)+W(IMP)*E(K3)
113 CONTINUE
    IDUM=IP(K1)
    W(IDUM)=T(2)
114 CONTINUE
```

This block of instructions multiplies the pricing vector by each of the pivot vectors in the eta file, this being done from the last pivot vector placed in the eta file to the first.

K1 is the number of the pivot vector. ME + 1-K6 allows the DO loop counter to drive K1 from the upper parameter of the loop to the lower. LU3 is the number of the first eta file entry in pivot vector K1. LU4 is the last eta file entry in pivot vector K1.

The inner DO loop multiplies the pricing vector by the pivot vector K1, element for element, and the products are summed and stored in T(2).

A pivot vector is identified by the row for which the column was pivoted. IP(I) then contains the pivot row number at pivot number I. As each pivot vector is multiplied by the pricing vector, the sum of the product (stored in T(2)) replaces the pricing vector value corresponding to the element IP(K1).

It should be noted that the pricing vector, after being multiplied by the eta file, represents a single row of the inverse matrix, or the sum of several rows multiplied by a factor. A row of the inverse matrix multiplied by the original values of the matrix of technological coefficients yields the current value of the matrix entry at that row column intercept. For example:

$$(A^{-1})^3 \cdot A_2 = \bar{a}_{2,3}$$

```
115 II=0
    HIA=0.0
    KM(22)=0
```

II, the number of the pivot column, is set to zero. HIA, the value of the largest positive coefficient of the objective function is set to zero, and KM(22), the number of eligible pivot columns, is initialized.

```
      DO 121 IZ=1,N
      IF(IKJ(IZ).NE.0) GO TO 120
      IF(JG(IZ).LT.KM(16)) GO TO 120
      MTA=KL(IZ)
      MTB=KL(IZ+1)-1
      T(2)=0.0
```

The DO statement begins a series of instructions which will multiply the original matrix by the pricing vector, column by column. If IKJ(IZ) is not zero, then column IZ is already in solution and the coefficient value will be zero; therefore multiplication by the pricing vector will be bypassed and the coefficient is later given the value of zero. The second IF statement checks to insure that the column is eligible as a pivot by its priority level. If the priority level of a column is less than the priority level being operated on, then the column is ineligible and the multiplication by the pricing vector is bypassed with the objective function coefficient for that column being set to zero in a later instruction.

MTA and MTB are established as the first and the last matrix entries in column IZ, respectively. T(2), whose value will later be the coefficient value of the objective function for column IZ, is set to zero.

```
      DO 116 K=MTA,MTB
      IN=IA(K)
      IF(ABS(W(IN)).LE.Z(4)) GO TO 116
      T(2)=T(2)+W(IN)*A(K)
  116 CONTINUE
```

This DO loop multiplies the original matrix by the pricing vector column by column. IN is the number of the row in which matrix entry K falls. The IF statement suppresses multiplication of pricing vector values which approach zero with a tolerance of Z(4). If the value is within the tolerance limits, A(K) (the matrix entry number K) is multiplied by the pricing vector value W(IN), and the product is summed with the other products.

```
        IF(JG(IZ).NE.KK) GO TO 118
        KB1=KB(IZ)
        IF(KN(IZ).EQ.POSD) GO TO 117
        T(2)=T(2)-WW(KB1)
        GO TO 118
    117 T(2)=T(2)-DY(KB1)
```

This instruction block finishes calculating the objective function coefficient value (traditionally Z_j-C_j). In goal programming, the C_j needs to be subtracted only if the objective function priority level is equal to the column priority level for which the coefficient is being calculated. The first IF statement in this block checks the priority level of the column variable, and if it is not equal to the priority level being operated on, the value of the coefficient T(2) is left as is. If not, KB1, the number of the original row which created the column IZ, is established. The second IF statement determines the type of deviation the column variable represents (POSD for a positive deviation and default for a negative deviation). If the column is a positive deviation, the differential weight, DY, associated with row KB1 is subtracted. If the column is a negative deviation, the differential weight, WW, associated with the original row, KB1, is subtracted.

```
    118 CAX(IZ)=T(2)
        IF(CAX(IZ).LE.Z(3)) GO TO 121
        KM(22)=KM(22)+1
        K6=KM(22)
        KNT(K6)=IZ
        IF(CAX(IZ).GT.HIA) GO TO 119
        GO TO 121
    119 HIA=CAX(IZ)
        II=IZ
        GO TO 121
    120 CAX(IZ)=0.0
    121 CONTINUE
```

This instruction block stores the objective function coefficients in array CAX, counts the number of eligible pivot columns, and chooses the largest value.

T(2), the sum of the products resulting from the multiplication of an original matrix column by the pricing vector, is the coefficient value of column IZ. This is stored in CAX. The first IF statement rejects columns as eligible pivot columns if the coefficient is less than or equal to the tolerance Z(3). If the coefficient is not rejected, KM(22), the number of eligible columns, is incremented. KNT contains the eligible pivot column. K6 represents that column number. The second IF statement compares the coefficient of the current column with the value of the highest positive coefficient value (HIA) chosen thus far.

If CAX(IZ) is greater than HIA, then HIA takes on the value of CAX(IZ) and the number of the column containing the larger coefficient is stored in II. The statement CAX(IZ)=0.0 is used for those columns which are already in solution, or whose priority level is lower (in numerical value) than the priority level being operated on.

```
    IF(KM(31).LE.0) GO TO 124
    WRITE(WRITEU,125)KK,(CAX(IK),IK=1,N)
125 FORMAT(1H0,35HOBJECTIVE FUNCTION ROW AT PRIORITY ,I5/1H ,4(1X,F1
   15.7))
```

If KM(31) is zero, then the preceding write statement is not executed.

```
    GO TO 124
122 CAX(II)=0.0
    HIA=0.0
    II=0
    K2=KM(22)
    DO 123 I=1,K2
    K6=KNT(I)
    IF(CAX(K6).LE.HIA) GO TO 123
    HIA=CAX(K6)
    II=K6
123 CONTINUE
124 RETURN
```

This last block of instructions is used only if a previously selected column has been rejected (IREJ is 1) and a new pivot column is to be selected.

The coefficient of the previously selected column is set to zero so that it will not be chosen again. The value of the largest positive coefficient (HIA) is again set to zero, and the number of the pivot column (II) is set to zero. The DO loop goes from 1 to the original number of eligible columns, KM(22). From those eligible columns the largest positive value and the number of that column are again chosen.

SUBROUTINE CNTRL (COMMON PARAMETERS) controls the execution of the modified revised simplex-product form of the inverse algorithm. While the actual execution takes place in other subroutines, the calls to the other subroutines are made in CNTRL. Bookkeeping functions, such as counting iterations, identifying the basis variables, checking of infeasibilities, etc., are also handled here. The flow chart for SUBROUTINE CNTRL is shown in Fig. 5.

```
IF(KM(36).EQ.0) GO TO 118
```

On entering CNTRL, this statement checks to insure that an initial solution has been set up with its associated computational columns (variables).

```
JJJ=0
```

JJJ is zero if no infeasibilities exist when best minimization of deviations from goal levels has been achieved. JJJ is 1 if this same point in execution is reached, but an infeasibility does exist.

```
101 CALL XCK(K,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,X,Y,
   1KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
```

XCK is called where infeasibilities are sought (negative right-hand-sides, or artificials in solution). K will be returned to CNTRL as 1 if an infeasibility does exist.

```
IF(KM(16).GT.MB) GO TO 106
```

XCK also searches for deviational variables in solution at the priority level being operated on (KM(16)). IF no deviation can be found, KM(16) is increased by 1 and a new search is initiated. Therefore, on leaving XCK, it is necessary to insure that KM(16) has not sought to be incremented beyond the number of priority levels which exist (MB). The inference here is that if KM(16) is greater than MB, then all possible deviations from goal levels have been driven from solution; therefore, an optimal solution has been achieved.

```
IF(KM(30)-K) 102,104,103
```

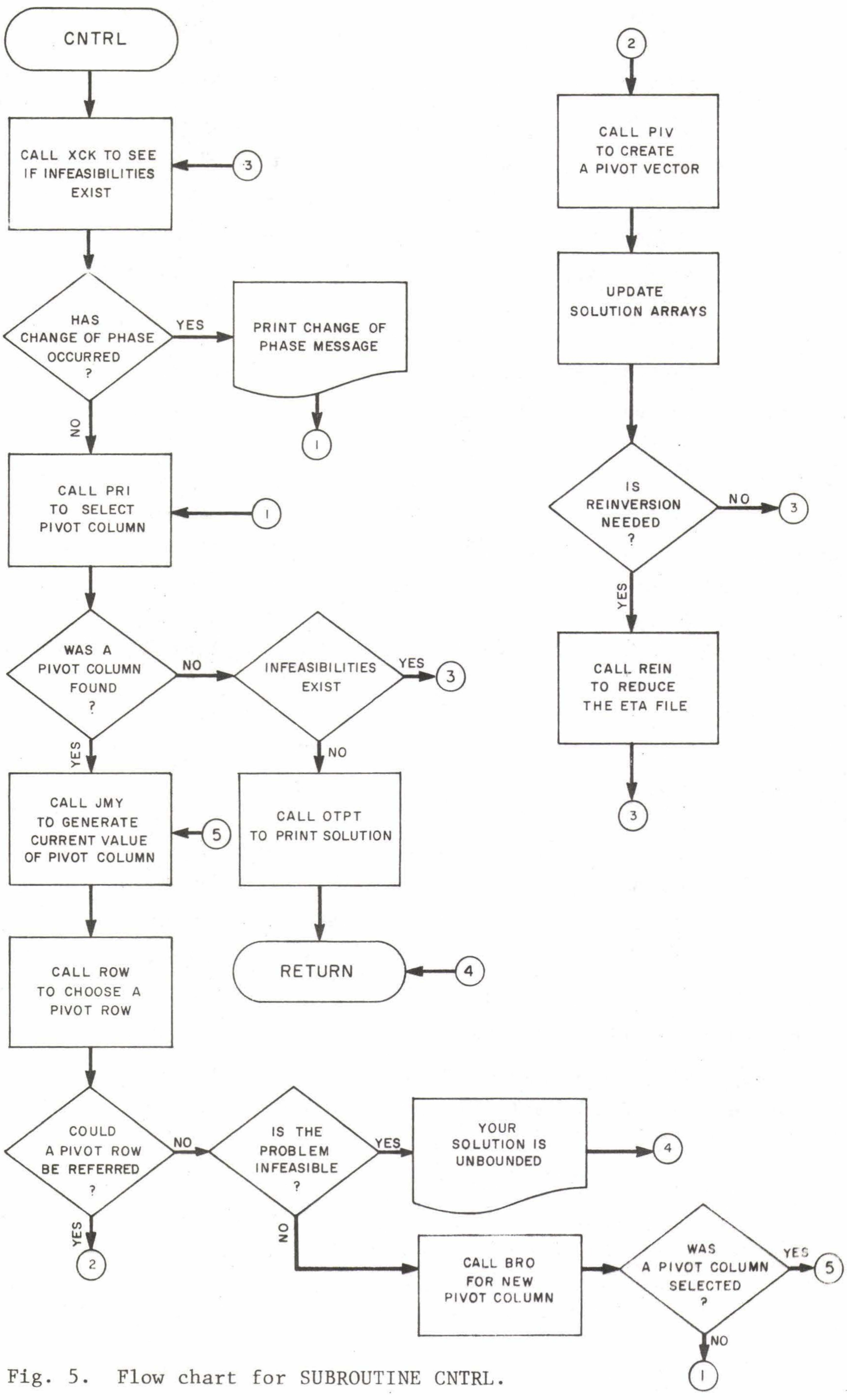
CNTRL
CALL XCK TO SEE IF INFEASIBILITIES EXIST
3
HAS CHANGE OF PHASE OCCURRED ?
YES
PRINT CHANGE OF PHASE MESSAGE
1
NO
CALL PRI TO SELECT PIVOT COLUMN
1
WAS A PIVOT COLUMN FOUND ?
NO
INFEASIBILITIES EXIST
YES
3
NO
YES
CALL JMY TO GENERATE CURRENT VALUE OF PIVOT COLUMN
5
CALL OTPT TO PRINT SOLUTION
RETURN
4
CALL ROW TO CHOOSE A PIVOT ROW
COULD A PIVOT ROW BE REFERRED ?
NO
IS THE PROBLEM INFEASIBLE ?
YES
YOUR SOLUTION IS UNBOUNDED
4
YES
2
NO
CALL BRO FOR NEW PIVOT COLUMN
WAS A PIVOT COLUMN SELECTED ?
YES
5
NO
1
2
CALL PIV TO CREATE A PIVOT VECTOR
UPDATE SOLUTION ARRAYS
IS REINVERSION NEEDED ?
NO
3
YES
CALL REIN TO REDUCE THE ETA FILE
3

Fig. 5. Flow chart for SUBROUTINE CNTRL.

This statement evaluates changes of phase. Phase I is when infeasibilities exist and phase II is when all infeasibilities have been resolved. If KM(30), the infeasibility flag, is 1, and XCK indicates that infeasibilities still exist (K is 1 also), then no change of phase occurs. But if KM(30) is 1 and K is 0, then KM(30) will be set to zero and a change of phase indicated.

```
102 KM(30)=1
    WRITE(WRITEU,121) KM(1)
121 FORMAT(30H *BECAME INFEASIBLE, ITERATION,I5)
    GO TO 104
```

These instructions are executed when KM(30) is 0 and K is 1 when control is returned from XCK.

```
103 KM(30)=0
    WRITE(WRITEU,122)KM(1)
122 FORMAT(23H FEASIBLE ON ITERATION ,I5)
```

These instructions are executed when KM(30) is 1 and K is 0 when control is returned from XCK.

```
104 CALL PRI(KLM,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    KKK=KM(22)
```

This CALL to PRI is a request for a pivot column to be found at a given priority level. KLM is the formal parameter which returns the number of the pivot column. KKK, at this point, is the number of eligible pivot columns.

```
    IF(KM(31).LE.0) GO TO 105
    WRITE(WRITEU,123)KLM
123 FORMAT(1H ,20HTHE PIVOT COLUMN IS ,I5)
```

KM(31) is non-zero if the option to write out the number of the pivot column is desired.

```
105 IF(KLM.EQ.0) GO TO 106
    GO TO 109
```

If the number of the pivot column, KLM, is zero, and an optimal solution has been reached, execution is sent to output procedures. If the pivot column is not zero, the GO TO sends execution to find the current value of the pivot column.

```
106 IF(KM(30).EQ.0) GO TO 107
    IF(JJJ.EQ.1) GO TO 107
    JJJ=1
    KM(16)=1
    GO TO 101
```

When an optimal solution is supposedly reached, a check is made to insure the solution is feasible (no negative right-hand-side values exist). If there are no infeasibilities (KM(30) is 0), execution is sent on to more termination and output procedures.

If infeasibilities still exist, and if an optimal solution has not been reached previously (JJJ is 0), then JJJ is set to 1 to indicate that an optimal solution has already been reached. KM(16), the priority level being operated on, is set to 1 and execution returns to the pivoting procedure in an attempt to drive out the infeasibility. This situation usually indicates that various tolerances are not eliminating rounding error residuals.

```
107 IF(ME.LT.3*M) GO TO 108
    ITYPE=3
    CALL REIN(ITYPE,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
```

Before the output routine is called, a check is made to see if the number of pivot vectors is less than three times the number of original rows (M). This is an arbitrary check designed to determine whether or not rounding errors might occur because of the many multiplications involved in the updating of a row or column. If ME is greater than or equal to 3xM, then a reinversion of the eta file is performed. If not, the output subroutine is called.

```
108 CALL OTPT(KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,X,Y,
   1KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    GO TO 120
```

The subroutine, which does the output functions related to the optimal solution, is called after which execution is returned to subroutine INPT where new commands are read and interpreted.

```
109 CALL JMY(KLM,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    CALL ROW(IR,KLM,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    IF(KM(31).LE.0) GO TO 110
    WRITE(WRITEU,124)IR
124 FORMAT(18H THE PIVOT ROW IS ,I5)
```

JMY is called to generate the current value of column number KLM. ROW is then called to find a pivot row, whose number will be returned through variable IR, from the pivot column KLM.

If KM(31) is greater than zero, the option for printing out the number of the pivot row is performed.

```
110 IF(IR.GT.0) GO TO 113
```

It is possible that subroutine ROW will not find a pivot row; therefore, IR is returned to CNTRL with a value of zero. This occurs if the column is rejected because a higher level priority level would be violated by introducing that column into the basis or if no suitable pivot row is found.

```
    KKK=KKK-1
    IF(KM(38).NE.0) GO TO 111
    IF(KKK.EQ.0) GO TO 119
111 KM(38)=0
```

If IR is returned from ROW as zero, it needs to be determined whether the column was rejected because of violation of a higher priority level, or because no suitable pivot row was found. Each time IR is returned as zero during an iteration, KKK (the number of eligible pivot columns remaining) is reduced by one. If all eligible columns fail to produce a pivot row when no upper priority level is violated (KM(38) is zero), then an unbounded solution exists.

```
    IREJ=1
    CALL BRO(KLM,IREJ,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
```

When a column is rejected because a higher priority is violated (IR is returned from ROW as zero and KM(38) as 1), IREJ is set to 1 and BRO is called. The IREJ of 1 will request that the list of eligible pivot columns be searched for the next highest objective function coefficient. When that column is found, its number is returned to CNTRL through KLM.

```
      IF(KLM.EQ.0) GO TO 112
      IF(KM(31).LE.0) GO TO 109
      WRITE(WRITEU,125) KLM
  125 FORMAT(1H ,20HNEW PIVOT COLUMN IS ,I5)
      GO TO 109
```

When KLM is returned from BRO, it is tested to see if it is the number of a column or zero. If KLM is the number of a column, an optional format can be written indicating the pivot column after which execution is sent back to JMY to calculate the current value of the column elements.

```
  112 IF(KM(16).EQ.MB) GO TO 106
      KM(16)=KM(16)+1
      GO TO 101
```

This set of instructions is executed if all eligible columns at an iteration are rejected. At that point, if KM(16) is equal to MB, then all priority levels have been operated on, so execution is sent to the output sequence. If all priority levels have not been operated on (KM(16) is less than MB), the priority level being operated on is incremented and execution is set back to XCK.

```
  113 KM(5)=NR(IR)
      CALL PIV(IR,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
     1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
```

When a pivot row and column have been identified, KM(5) is set to the name of the pivot row, and PIV is called to create an eta vector (pivot vector).

```
      IKJ(KLM)=IR
      IJ=JH(IR)
      IKJ(IJ)=0
      JH(IR)=KLM
      KM(6)=KN(KLM)
```

After a pivot has been performed, the arrays which keep track of the solution are adjusted, and KM(6) is set to the name of the pivot column.

```
KM(1)=KM(1)+1
KM(17)=KM(17)+1
```

KM(1), the number of iterations, and KM(17), the number of iterations since last reinversion, are incremented.

```
KK=KM(10)
AKK=FLOAT(KK)
AKKK=AKK/(FLOAT(M)*FLOAT(ME))
IF(AKKK.GT.0.80) GO TO 116
```

These instructions calculate the size of the eta file (all of the pivot vectors) and the present density of non-zero entries. If this density AKKK exceeds .80, a reinversion of the eta file is performed.

```
IF(KM(13).EQ.1) GO TO 114
IF(KM(1).GT.KM(4)) GO TO 117
IF(KM(17).GT.KM(29)) GO TO 114
GO TO 101
```

The first instruction checks to see if reinversion is needed because the memory allocated to eta file entries is exceeded. The second instruction checks to see if the number of iterations, KM(1), is exceeded by the number of iterations allowed, KM(4). The third instruction checks the number of iterations since the last reinversion, KM(17), with the maximum number of iterations allowed between reinversions, KM(29).

```
114 ITYPE=2
115 CALL REIN(ITYPE,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,X,
   1Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    GO TO 101
```

ITYPE is set to 2 when reinversion is required because the memory allocation for the eta file is exceeded. REIN is called to perform any type of reinversion.

```
116 ITYPE=1
    GO TO 115
```

ITYPE is set to 1 when reinversion is required because density of the eta file causes inefficiencies in multiplying through the eta file. The GO TO sends execution to call REIN.

```
117 WRITE(WRITEU,126)
126 FORMAT(1H0,48HEXECUTION TERMINATED BECAUSE OF ITERATION LIMIT.)
    GO TO 120
```

This diagnostic is printed when the number of iterations executed in the problem exceeds the maximum number of iterations allowed.

```
118 WRITE(WRITEU,127)
127 FORMAT(1H0,71HEXECUTION CANNOT PROCEED UNTIL THE INITIAL FEASIBLE
   1 SOLUTION IS SET UP./1H ,35HA SNGL OR SOLV COMMAND IS REQUIRED.)
    GO TO 120
```

This diagnostic is printed when subroutine CNTRL is entered before the computational variables necessary for an initial solution have been created by subroutine ART.

```
119 WRITE(WRITEU,128)
128 FORMAT(1H0,27HYOUR SOLUTION IS UNBOUNDED.)
120 RETURN
```

This diagnostic is printed when the solution is feasible and a pivot row cannot be found in the indicated pivot columns (all eligible pivot columns at that priority level).

SUBROUTINE INPT (COMMON PARAMETERS) performs the task of reading or processing the commands, processes all functions of the CHNG command, and directs execution to the proper subroutines required by a given command. The flow chart for SUBROUTINE INPT is shown in Fig. 6.

```
    KKK=0
    KERR=1
```

KKK is zero when a PARA command is given and an advanced basis does not exist. KKK is 1 after the first run of the parametric series, indicating that the basis solution is available for advanced basis starts.

KERR is the variable which controls the computed GO TO statements following calls to subroutines ART, REIN and CNTRL. This device is used because the subroutines can be called by several command sequences; therefore, KERR is needed to direct execution back to the instruction sequence which initiated the call.

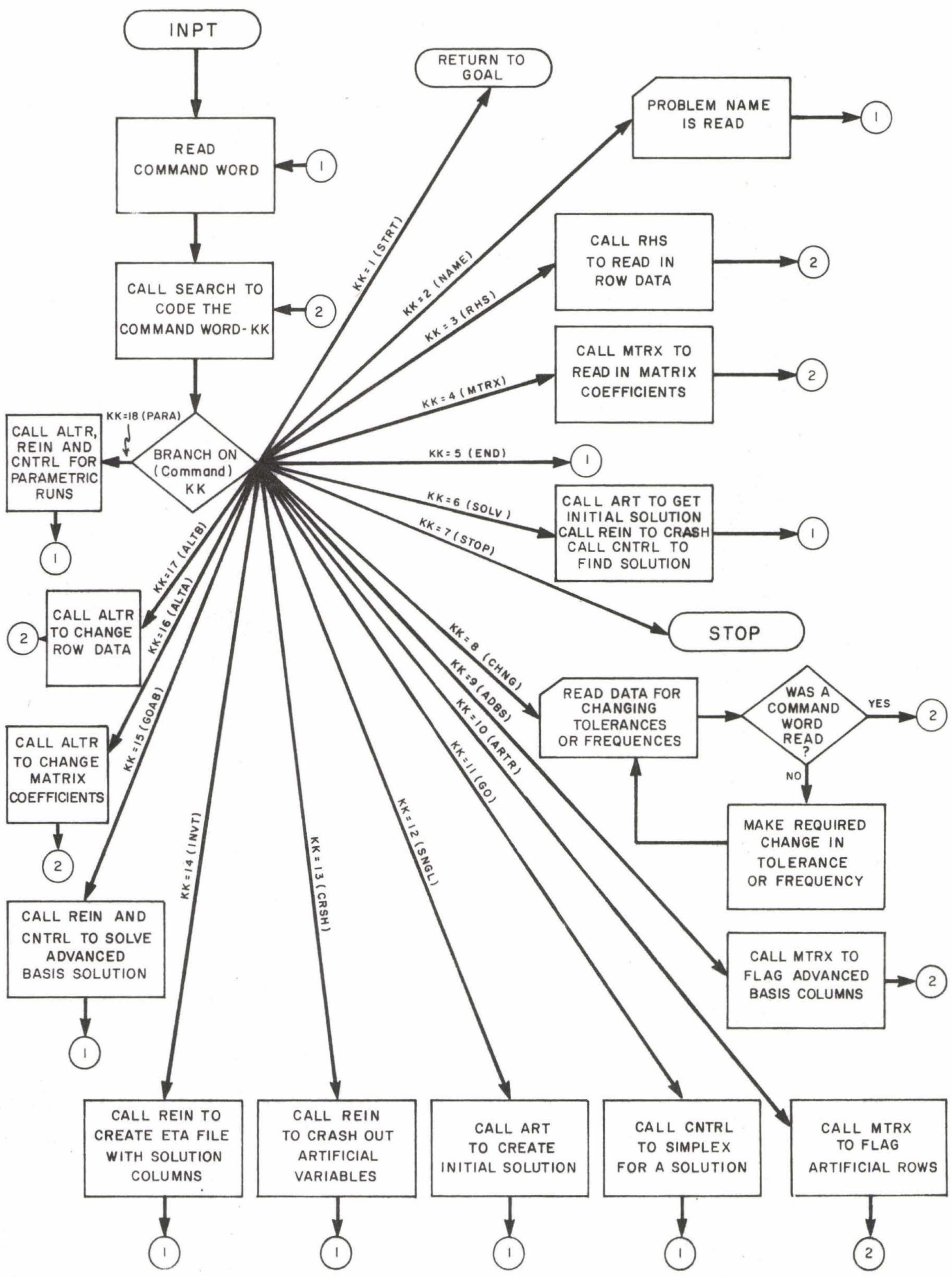
INPT
READ COMMAND WORD
CALL SEARCH TO CODE THE COMMAND WORD-KK
BRANCH ON (Command) KK
RETURN TO GOAL
PROBLEM NAME IS READ
CALL RHS TO READ IN ROW DATA
CALL MTRX TO READ IN MATRIX COEFFICIENTS
CALL ART TO GET INITIAL SOLUTION CALL REIN TO CRASH CALL CNTRL TO FIND SOLUTION
STOP
READ DATA FOR CHANGING TOLERANCES OR FREQUENCES
WAS A COMMAND WORD READ ?
YES
NO
MAKE REQUIRED CHANGE IN TOLERANCE OR FREQUENCY
CALL MTRX TO FLAG ADVANCED BASIS COLUMNS
CALL ALTR, REIN AND CNTRL FOR PARAMETRIC RUNS
CALL ALTR TO CHANGE ROW DATA
CALL ALTR TO CHANGE MATRIX COEFFICIENTS
CALL REIN AND CNTRL TO SOLVE ADVANCED BASIS SOLUTION
CALL REIN TO CREATE ETA FILE WITH SOLUTION COLUMNS
CALL REIN TO CRASH OUT ARTIFICIAL VARIABLES
CALL ART TO CREATE INITIAL SOLUTION
CALL CNTRL TO SIMPLEX FOR A SOLUTION
CALL MTRX TO FLAG ARTIFICIAL ROWS
KK=1 (STRT)
KK=2 (NAME)
KK=3 (RHS)
KK=4 (MTRX)
KK=5 (END)
KK=6 (SOLV)
KK=7 (STOP)
KK=8 (CHNG)
KK=9 (ADBS)
KK=10 (ARTR)
KK=11 (GO)
KK=12 (SNGL)
KK=13 (CRSH)
KK=14 (INVT)
KK=15 (GOAB)
KK=16 (ALTA)
KK=17 (ALTB)
KK=18 (PARA)
1
2

Fig. 6. Flow chart for SUBROUTINE INPT.

```
101 READ(READU,142)K,(KBCD(I),I=3,21)
142 FORMAT(20A4)
102 WRITE(WRITEU,143)K
143 FORMAT(1H0,A4)
```

A card is read with K storing the first four characters which constitute the command, and KBCD(3-21) contain the remainder of the card image. The WRITE statement echoes the command word.

```
    CALL SEARCH(K,KK)
    IF(KK.EQ.100) GO TO 101
    GO TO (141,103,104,105,101,107,117,118,124,126,114,127,128,
   1129,130,132,136,137),KK
```

K, which contains the command word, is sent to subroutine SEARCH, where variable KK is given a number which will direct control to the proper statement label of the computed GO TO.

The IF statement is used to direct control to a diagnostic if the command word in K is not in the command vocabulary of SEARCH. KK is set to 100 if this situation occurs.

```
103 READ(READU,144) (PRONAM(I),I=1,10),KBCD(29),KBCD(30)
144 FORMAT(10A4,32X,2A4)
    WRITE(WRITEU,145)(PRONAM(I),I=1,10)
145 FORMAT(1H ,10A4)
    GO TO 101
```

Execution of this block of instructions takes place only if the command word NAME is read. PRONAM will store the 40 characters allowed in the problem name, and the WRITE statement will echo the name.

```
104 CALL RHS(K,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    GO TO 102
```

Subroutine RHS is called with the purpose of entering data pertinent to rows of the matrix. Row names, right-hand-side values, types of inequalities, priority levels and weights, and row descriptions are handled through RHS. The argument K returns to INPT the command word which terminated execution in RHS.

```
105 KK=1
106 CALL MTRX(K,KK,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    GO TO 102
```

A call to subroutine MTRX can accomplish three purposes:

(i) reading in the matrix coefficient values by column and row (command word is MTRX, KK is 1)

(ii) flagging solution columns in an advanced basis start (command word is ADBS, KK is 2), and

(iii) flagging rows which contain an artificial variable in solution for an advanced basis start (command word is ARTR, KK is 3).

```
107 KK=6
    IF(KBCD(3).EQ.BLANK) GO TO 134
    IF(KBCD(3).EQ.MEDM) GO TO 108
    KM(35)=3
    GO TO 134
```

The command word SOLV causes execution of this block of instructions. KBCD(3) contains the card image of columns 5-8 of the command card. In conjunction with SOLV, these columns determine the type of output desired:

(i) blank - long output,

(ii) MEDM - medium output, or

(iii) non-blank and not MEDM - short output.

KM(35) is either 1, 2, or 3, respectively.

The GO TO instruction sends execution to a block of initialization instructions.

```
108 KM(35)=2
    GO TO 134
```

KM(35) is set to 2 because a GO or SOLV card had a MEDM in columns 5-8. The GO TO sends execution to the block of initialization instructions.

```
109 CALL ART(KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    KKK=0
    GO TO (110,113,113,138,135,133),KERR
```

ART is called to create an initial solution and the computational variables required by it. KKK is set to zero indicating that an advanced basis solution does not exist. The GO TO sends control back to the instruction sequence which required ART.

```
110 KM(33)=1
    ITYPE=4
```

KM(33) is set to 1, indicating that feasible crashing is desired when REIN is called. ITYPE is 4 when feasible crashing occurs.

```
111 CALL REIN(ITYPE,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    GO TO (112,113,131,140),KERR
```

The call to REIN is made whenever reinversion of the eta file is needed, or feasible crashing is desired. INVT is the command for reinversion alone, and CRSH is the feasible crashing command. This call to REIN is also implicit in the instruction sequences of PARA, SOLV, and GOAB commands.

```
112 CALL CNTRL(KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    KKK=1
    GO TO (113,113,133),KERR
```

CNTRL is called most directly by the command GO. It is also implicit in the SOLV, PARA, and GOAB commands.

KKK is set to 1 indicating that there now exists a basis solution suitable for advanced basis starts.

The GO TO directs execution back to the instruction sequence which required CNTRL.

```
113 KERR=1
    GO TO 101
```

KERR is nominally 1; therefore, when operations which require other values are completed, KERR is again set to 1.

```
114 LE(1)=1
    DO 115 I=1,M
115 X(I)=B(I)
    IF(KBCD(3).EQ.BLANK) GO TO 112
    IF(KBCD(3).EQ.MEDM) GO TO 116
    KM(35)=3
    GO TO 112
116 KM(35)=2
    GO TO 112
```

The command word GO causes execution of this block of instructions. LE(1) is set to 1, indicating that the problem solving process has begun. The DO loop initializes the current values for the right-hand-sides (X(I)), with the original values for the right-hand-sides (B(I)).

The IF statements check columns 5-8 of the command card (stored in KBCD(3)) for blank, MEDM, or non-blank and non-MEDM. This indicates that a long output, medium output, or short output is desired, respectively. Accordingly, KM(35) is set to 1, 2, or 3, respectively.

```
            117 STOP 7
```

This statement is executed only if the command word STOP is encountered.

```
     (i)    118 READ(READU,146) K,KBCD(2),III,AA
            146 FORMAT(2A4,I1,F16.8)
    (ii)        IF(K.NE.KM(25)) GO TO 102
                WRITE(WRITEU,147) KBCD(2),III,AA
            147 FORMAT(1H0,4X,A4,I1,1X,F16.8)
   (iii)        IF(KBCD(2).EQ.TOLR) GO TO 121
                IF(KBCD(2).NE.FREQ) GO TO 122
    (iv)        IF(III.GT.2) GO TO 123
                II=IFIX(AA)
                GO TO (119,120),III
            119 KM(29)=II
                GO TO 118
            120 KM(4)=II
                GO TO 118
     (v)    121 IF(III.GT.8) GO TO 123
                IF(III.GT.6) GO TO 123
                Z(III)=AA
                GO TO 118
    (vi)    122 WRITE(WRITEU,148) KBCD(2)
            148 FORMAT(1H0,A4,53H IS NOT A LEGITIMATE COMMAND TO FOLLOW THE CHNG
               1 CARD.)
                STOP
            123 WRITE(WRITEU,149) III,KBCD(2)
            149 FORMAT(1H0,I1,31H IS AN IMPROPER NUMBER FOR THE ,A4,6H CARD.)
                STOP
```

This block of instructions is executed by the command word CHNG. Tolerance and frequency values can be changed.

(i) Reads the data cards following the CHNG command. K will contain any command words in columns 1-4, KBCD(2) contains the description of the type of change to be effected; TOLR for tolerance change, or FREQ for frequency changes, III contains the number which indicates the particular tolerance or frequency to be changed, and AA contains the value to which the tolerance or frequency is to be changed.

(ii) Check K for a blank, if non-blank, K is sent to subroutine SEARCH to be interpreted.

(iii) This IF statement checks to see if changes are for tolerance or frequencies.

(iv) If frequencies are being changed, any particular frequency indicated by III cannot exceed 2. This is because only two frequencies can be changed. The GO TO statement directs execution to the proper variable which contains the frequency value. III is 1 when a change is made on the reinversion frequency, and 2 for changes in maximum number of iterations allowed. The variables KM(29) and KM(4), respectively, contain those frequencies.

(v) This starts the section where tolerance changes are made. Only tolerances 1, 2, 3, 4, 5, and 8 can be changed. The IF statement insures that these are the only values III has taken on. The variable Z is used to store tolerances, and III is used as the index to change those values.

(vi) These are the format statements which give diagnostics when the number of the index III is not correct and when TOLR or FREQ are not used as change commands.

```
124 KK=7
    GO TO 134
125 KK=2
    GO TO 106
```

These instructions are executed by the ADBS command word. First ART is called to create the initial solution through the initialization instruction block, then MTRX is called to flag the advanced basis columns.

```
126 KK=3
    GO TO 106
```

KK is set to 3 indicating that rows with artificials are to be flagged when MTRX is called. The GO TO calls MTRX.

```
127 KK=5
    GO TO 134
```

These instructions are executed only on the SNGL command. Certain variables are initialized and the initial basis is created by ART.

```
128 KERR=2
    GO TO 110
```

These instructions are executed only by the CRSH command. The GO TO sends execution to the CALL REIN sequence, and KM(33) is set to 1.

```
129 ITYPE=5
    KERR=2
    GO TO 111
```

These instructions are executed only by the INVT command. ITYPE is set to 5 to indicate a reinversion which will install the requested advanced basis columns. The GO TO calls REIN.

```
130 KERR=3
    ITYPE=5
    GO TO 111
131 KERR=1
    GO TO 110
```

These instructions are executed by the GOAB command. These instructions are used after the command sequence ADBS, END, ARTR, END. GOAB is equivalent to the command sequence INVT, CRSH, and GO.

```
132 KK=1
133 CALL ALTR(K,KK,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
```

KK is set to 1 before ALTR is called indicating that a data change is to be made on a matrix coefficient value. ALTR is capable of making data changes in any data cards which follow the RHS or MTRX command, and also controls parametric runs.

```
    IF(KM(34).EQ.0) GO TO 135
```

When execution leaves ALTR, KM(34) is checked for zero. If KM(34) is zero, it indicates that no changes have been made in ALTR which would require that the initial solution be recreated. If KM(34) is not zero, the block of instructions used for initializing certain variables before ART is called is executed.

```
134 N=N-KM(14)-KM(26)
    MA=MA-KM(14)-KM(26)
    KM(1)=0
    KM(14)=0
    KM(26)=0
    KM(8)=0
    ME=0
    KM(16)=0
    KM(30)=0
    KM(34)=0
    KERR=5
    GO TO 109
135 KERR=1
    GO TO(102,102,138,138,101,110,125),KK
```

This is the block of instructions used to bring the matrix back to the original data which was entered (i.e. without computational columns, etc.). The first GO TO calls ART which will recreate the initial solution.

```
136 KK=2
    GO TO 133
```

KK is set to 2 before the GO TO sends execution to the call to ALTR, indicating that changes are to be made in RHS data.

```
  (i)     137 KK=3
              IF(KM(36).EQ.1) GO TO 133
              KKK=0
              KERR=6
              GO TO 109
 (ii)     138 IF(KKK.EQ.1) GO TO 139
              KKK=1
              KERR=4
              GO TO 110
(iii)     139 ITYPE=5
              KERR=4
              GO TO 111
          140 KERR=3
              GO TO 112
```

This last block of instructions is initiated by the PARA command.

(i) KK is set to 3 to indicate a parametric run. KM(36) is 1 only if an initial solution has been created. If this is the case, ALTR is called and a parametric run begins. If not, ART is called to create the initial solution. KKK is set to zero to indicate that no advanced basis solution exists of an advanced basis start.

(ii) This is the instruction to which execution is returned after ALTR has been left and if KK is 3. KKK is checked to determine if an advanced basis exists. If so (KKK is 1), execution goes to (iii).

(iii) This is where a reinversion is performed and the advanced basis columns are installed, after which CNTRL is called to minimize the deviations from the goal levels. If KKK is zero, then REIN is called for feasible crashing before CNTRL is called.

```
      141 RETURN
```

This instruction is executed only if the command word STRT is read. Control is returned to the main program where the array and variable initialization process takes place.

SUBROUTINE JMY (J, COMMON PARAMETERS) performs a multiplication on the original value of a single column by each of the pivot vectors in the eta file. The flow chart for SUBROUTINE JMY is shown in Fig. 7.

Unique formal parameters:

J - is the number of the column whose current value will be calculated.

```
          KM(6)=KN(J)
```

KM(6) is set to the name of the column stored in the current value vector Y(I).

```
          DO 101 I=1,M
      101 Y(I)=0.0
```

This DO loop sets the current value vector, Y(I), to zero.

```
          LUMP=KL(J)
          LUMP1=KL(J+1)-1
          DO 102 K=LUMP,LUMP1
          IMP=IA(K)
          Y(IMP)=A(K)
      102 CONTINUE
```

LUMP and LUMP1 are the first and last non-zero matrix entries, respectively, in column J. The DO loop sets the current value vector to the original value of column J. IA(K) is the number of the row in which matrix entry number K

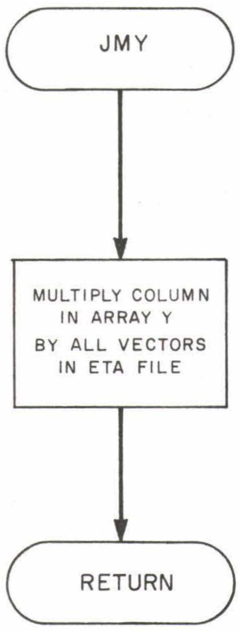

Fig. 7. Flow chart for SUBROUTINE JMY.

falls and IMP takes on this value. A(K) is the value of the matrix entry number K. In this manner only those elements of Y which are non-zero need to be operated on, the other values of Y(I) already being set to zero.

```
IF (ME) 103,107,103
```

ME is the number of pivot vectors formed. If ME is zero then the original value of column J which is stored in vector Y is also the current value. In this case Y(I) is returned to CNTRL as is. If ME is not zero, then Y(I) will be multiplied by ME pivot vectors.

```
(i)     103 DO 106 I=1,ME
            IDIM=IP(I)
            DT=Y(IDIM)
            Y(IDIM)=0.0
            IF(ABS(DT)-Z(4)) 106,106,104
        104 LUMP2=LE(I)
            LUMP3=LE(I+1)-1
(ii)        DO 105 K=LUMP2,LUMP3
            II=IE(K)
            Y(II)=Y(II)+E(K)*DT
        105 CONTINUE
        106 CONTINUE
        107 RETURN
```

(i) This DO loop goes through each of the pivot vectors which have been formed. IP is the number of the row which was pivoted out at pivot number I; therefore, DT is the value of the current storage vector at that corresponding row. The IF statement insures that the pivot element DT is sufficiently positive or negative so as not to consider its value zero.

(ii) This block of code is used when the pivot element is determined to be non-zero. LUMP2 and LUMP3 are the first and the last pivot entries in pivot vector number I, respectively. At this point each element of Y is updated by adding its current value to the product of the current pivot element in Y times the element of the pivot vector which corresponds to the Y element being updated.

For example: in standard matrix multiplication, a pivot element would be

$$E = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{bmatrix}$$

where a single column is non-identity. In revised simplex, only the non-identity vector would be stored

$$E_2 = \begin{bmatrix} 2 \\ 5 \\ 4 \\ -2 \end{bmatrix}$$

with the column of the pivot matrix being identified by IP. In this case IP(I)=2 where I is the number of the pivot which created the vector.

Multiplying the original value of a column Y of the matrix by the pivot matrix in standard matrix operation, we get

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{bmatrix} \quad x \quad \begin{bmatrix} 1 \\ 4 \\ 5 \\ 3 \end{bmatrix} \quad = \quad \begin{bmatrix} 9 \\ 20 \\ 21 \\ -5 \end{bmatrix}$$

where:

$$\begin{aligned} (1x1) + (2x4) + (0x5) + (0x3) &= 9 \\ (0x1) + (5x4) + (0x5) + (0x3) &= 20 \\ (0x1) + (4x4) + (1x5) + (0x3) &= 21 \\ (0x1) + (-2x4) + (0x5) + (1x3) &= -5 \end{aligned}$$

Observe that the only non-zero contributions to the totals come from the non-zero entries in the matrix E. These are of two kinds:

(i) The entries in column 2 of E, which are multiplied by entry number 2 of Y (which is 4 in our example), and

(ii) The 1's on the diagonal of E (other than in column 2), which are multiplied by the corresponding entries in Y (other than entry number 2).

Therefore in revised simplex the value of entry number 2 of Y (equal to 4) is set aside as DT and the matrix multiplication takes on the simplified form:

$$\begin{bmatrix} 2 \\ 5 \\ 4 \\ -2 \end{bmatrix} x4 \ + \begin{bmatrix} 1 \\ 0 \\ 5 \\ 3 \end{bmatrix} = \begin{bmatrix} 9 \\ 20 \\ 21 \\ -5 \end{bmatrix}$$

where:

$$\begin{aligned} (2x4) + 1 &= 9 \\ (5x4) + 0 &= 20 \\ (4x4) + 5 &= 21 \\ (-2x4) + 3 &= -5 \end{aligned}$$

which gives considerable savings in computational effort.

SUBROUTINE MTRX (KK, COMMON PARAMETERS) provides for the entry of the matrix coefficient data, flagging of columns for advanced basis starts, and flagging of rows which contain artificial variables for advanced basis starts. The flow chart for SUBROUTINE MTRX is shown in Fig. 8.

Unique formal parameters:

KK - directs execution to enter matrix coefficients, column flagging, or row flagging. The value of KK will be 1, 2, or 3, respectively.

```
      GO TO (101,115,122),KK
```

The GO TO directs execution to the matrix coefficient entry process, flagging advanced basis columns, or flagging artificial rows if KK is 1, 2, or 3, respectively.

```
  101 IF(KM(37).LE.0) GO TO 102
      WRITE(WRITEU,125)
  125 FORMAT(1H0,3X,40HCOL.     ROW          INTERSECT        COLUMN/1H
     1 ,3X,44HNAME     NAME            VALUE          DESCRIPTION/1H )
```

KM(37) will be non-zero if the option for writing out the matrix data as it is read into the program is desired.

```
  102 IF(M.EQ.0) GO TO 103
      GO TO 104
```

If M is equal to zero, then the RHS data has yet to be read and a diagnostic WRITE statement is called.

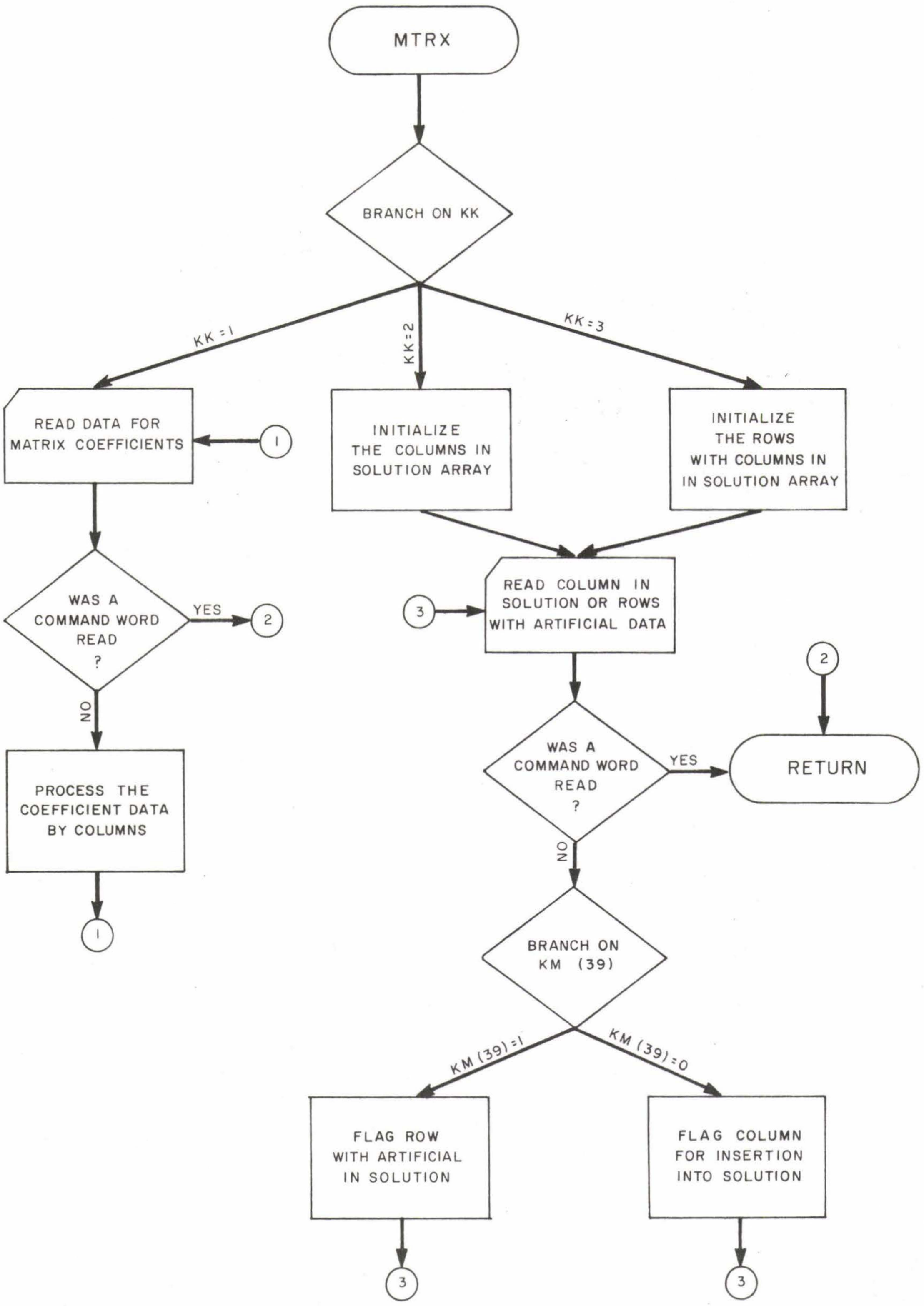
MTRX
BRANCH ON KK
KK=1
KK=2
KK=3
READ DATA FOR MATRIX COEFFICIENTS
1
INITIALIZE THE COLUMNS IN SOLUTION ARRAY
INITIALIZE THE ROWS WITH COLUMNS IN IN SOLUTION ARRAY
WAS A COMMAND WORD READ ?
YES
2
NO
PROCESS THE COEFFICIENT DATA BY COLUMNS
1
3
READ COLUMN IN SOLUTION OR ROWS WITH ARTIFICIAL DATA
2
WAS A COMMAND WORD READ ?
YES
RETURN
NO
BRANCH ON KM (39)
KM (39)=1
KM (39)=0
FLAG ROW WITH ARTIFICIAL IN SOLUTION
FLAG COLUMN FOR INSERTION INTO SOLUTION
3
3

Fig. 8. Flow chart for SUBROUTINE MTRX.

```
103 WRITE(WRITEU,126)
126 FORMAT(1H ,49HTHE RIGHT-HAND-SIDE VALUES MUST BE ENTERED BEFORE
   1/37H THE TECHNOLOGICAL MATRIX IS ENTERED.)
    STOP 7
```

This diagnostic is written when subroutine MTRX has been called with KK equal to 1 and M equal to zero. This indicates that the RHS data has not yet been read.

```
104 READ(READU,127)K,JCOL,IROW,AA,(KBCD(II),II=1,5)
127 FORMAT(3A4,F12.0,36X,5A4)
```

This READ statement enters the matrix coefficient data. K is any command word entered in columns 1-4. JCOL is the name of the column in which the coefficient lies. IROW is the name of the row in which the coefficient lies. AA is the matrix coefficient entered on that card. KBCD(1-5) is a 20-character description of what the column represents.

```
    IF(K-KM(25)) 124,105,124
```

K, which will contain any command word read, is checked for being blank. If so, then processing continues. If not, execution is returned to INPT.

```
105 IF(KM(37).LE.0) GO TO 106
    WRITE(WRITEU,128)JCOL,IROW,AA,(KBCD(II),II=1,5)
128 FORMAT(1H ,3X,2(A4,4X),F12.3,4X,5A4)
```

KM(37) is non-zero if the option to echo the card image of the MTRX data cards is desired.

```
106 DO 107 IJ=1,M
    II=IJ
    IF(NR(IJ)-IROW)107,108,107
107 CONTINUE
    WRITE(WRITEU,129)IROW,JCOL,IROW
129 FORMAT(1H0,A4,60H DOES NOT AGREE WITH A ROW NAME GIVEN AFTER THE R
   1HS COMMAND./1H ,60HCHECK FOR A FIELD SHIFT OR MISSPELLING IN MATRI
   2X ENTRY CARD ,2A4)
    ERFLG=1
    GO TO 104
```

The DO loop searches all of the row names entered after the RHS command, and establishes the number of the row associated with the row name. If the search can find no match between a row name submitted in the MTRX data, and one previously entered in a RHS data card, the above diagnostic is written.

```
108 MA=MA+1
    IF(MA.LE.NNN) GO TO 109
    WRITE(WRITEU,132) MA,NNN
132 FORMAT(29H0 THE NUMBER OF MTRX ENTRIES ,I5,33H EXCEEDS NUMBER ON
   1PARAMETER CARD ,I5/1H ,4X,59HNUMBER OF MTRX CARDS SHOULD AGREE WI
   2TH THE FOURTH PARAMETER/23H OF THE PARAMETER CARD.)
    ERFLG=1
    GO TO 104
```

When the row number has been established, MA (the number of matrix entries) is increased by one. A check is made then for the number of MTRX cards. If the number is larger than the maximum entered on the parameter card, then an error message is written.

```
109 IF(N)111,111,110
```

A check is made to see if N, the number of columns, is zero. This is necessary because the coefficients are entered column by column. A check will later be made to see if the change from one column to the next has been made; therefore, when N is zero that check need not be made.

```
110 IF(KN(N)-JCOL)111,113,111
```

KN(N) is the name of the last column created. This is checked against JCOL, the name of the last data column read, to see if a new column is to be created. This is made necessary by the column by column approach to matrix data entry.

```
111 N=N+1
    IF(N-NB)112,112,114
112 KL(N)=MA
    KN(N)=JCOL
    JG(N)=999
    KB(N)=8888
```

When it is noted that the name of the last column read differs from the name of the next to the last column read, N is incremented by one, indicating the creation of a new column. The IF statement insures that the number of columns created does not exceed the number of columns input on the parameter card.

KL(N) is the number of the matrix entry which is the first matrix entry of column N. KN(N) is the name associated with column N. JG(N) is the priority level of column N and since there is no priority level associated with the decision variables in goal programming, 999 is the code for no priority level. KB(N) is the number of the constraint row which required the creation of column N. Since decision variables were not created on the requirement of a row, 8888 is given as the value to indicate this situation.

```
    III=(M+N)*5
    LABL(III-4)=KBCD(1)
    LABL(III-3)=KBCD(2)
    LABL(III-2)=KBCD(3)
    LABL(III-1)=KBCD(4)
    LABL(III)=KBCD(5)
```

All row and column descriptions (20 characters) are stored in array LABL. Five words of LABL are required to store the 20 characters. All row labels are stored first; therefore, the labels for column 1 end in word (M+1)*5 of array LABL.

The above block of instructions employs this storage technique to store the column descriptions. It should be noted that only the data card beginning each column need contain the column description. This is due to the column by column matrix entry process.

```
113 IA(MA)=II
    A(MA)=AA
    GO TO 104
```

When the row and column numbers of a matrix entry are established, the row associated with matrix entry number MA is stored in IA. The value of matrix entry number MA is AA.

```
114 WRITE(WRITEU,130)
130 FORMAT(1H ,53HTHE NUMBER OF COLUMNS IN COEFFICIENT MATRIX EXCEEDED
   1,/39H CHECK FOR MISSPELLING OF COLUMN NAMES.)
    ERFLG=1
    GO TO 104
```

This is the diagnostic written when the number of input columns is exceeded.

```
115 DO 116 I=1,N
116 IKJ(I)=0
    KM(39)=0
```

This instruction sets the values of all columns in row solution, IKJ, to zero. KM(39) is set to zero indicating that the data which will be read will be used to establish the numbers of the columns which will be used in advanced basis starts.

```
(i)     117 READ(READU,131) K,(KBCD(I),I=3,13)
        131 FORMAT(A4,11I4)
(ii)        IF(K-KM(25))124,118,124
(iii)   118 DO 121 I=3,13
            IF(KBCD(I).EQ.0) GO TO 117
            JJ=KBCD(I)
            IF(KM(39))120,119,120
        119 IKJ(JJ)=1
            GO TO 121
        120 JH(JJ)=0
        121 CONTINUE
            GO TO 117
```

(i) The READ statement gives K the command word which may be contained in columns 1-4 of the data card. KBCD contains 11 fields of 4 alphanumeric characters which will be the numbers of the columns desired for advanced basis starts, or the numbers of the rows which have artificials in solution.

(ii) After the READ, K is checked for values other than blanks; if a non-blank exists, control is returned to INPT.

(iii) This DO loop processes the 11 entries taken from each card. Each field is checked for a zero, and if a zero is encountered, control is sent to the READ statement to read a command word.

KM(39) is evaluated for either a zero or 1. If it is zero, then IKJ(JJ)=1 is executed, indicating column JJ is an advanced basis column. If it is 1, then JH(JJ)=0 is executed, indicating that row JJ contains an artificial variable.

```
122 DO 123 I=1,M
123 JH(I)=12345
    KM(39)=1
    GO TO 117
124 IF (ERFLG.GE.1) STOP
    RETURN
```

This instruction set initializes all columns in row solution to 12345, indicating that an artificial variable is not in solution in row I. KM(39) is set to 1, indicating that the READ statement will be reading the numbers of rows which have artificials in solution.

SUBROUTINE OTPT (COMMON PARAMETERS) produces all output when an optimal solution is found, and produces a listing of the numbers of the columns which are in solution and the number of the rows which have artificials in solution. The flow chart for SUBROUTINE OTPT is shown in Fig. 9.

```
    IF (KM(35).EQ.4) GO TO 108
    JIJ=0
    MTB=N
```

The IF statement checks for the parametric run output option. JIJ is zero when columns in solution are being written out, and 1 when the row numbers with artificials in solution are being determined. MTB is the upper parameter of the DO loop which will go through the columns and the rows. Here it is set to N, the number of columns, to determine the solution columns.

```
    WRITE(WRITEU,139)
139 FORMAT(1H0,49HTHIS SECTION USED ONLY FOR ADVANCED BASIS STARTS./
   11H ,49HTHESE ARE THE NUMBERS OF THE COLUMNS IN SOLUTION.)
```

This is a heading to identify advanced basis columns and rows.

```
101 JJ=0
    MTA=1
```

JJ keeps track of the number of columns found in solution, or rows with artificials. When JJ is 11, output occurs. MTA is the lower parameter of the DO loop.

```
102 DO 105 J=MTA,MTB
    JJJ=J
    IF(JIJ.EQ.1) GO TO 103
    IF(IKJ(J).EQ.0) GO TO 105
    GO TO 104
103 IF(JH(J).NE.0) GO TO 105
104 JJ=JJ+1
    KBCD(JJ)=J
    IF(JJ.EQ.11) GO TO 106
105 CONTINUE
```

This block of instructions searches all columns or all rows. JJJ is the value of the loop index when the loop is left. This will be used to calculate the value of MTA when the search through the rows or columns is continued. This feature is necessitated by the output procedure of using 11 fields of 4 alphanumeric characters, and using the array KBCD as the general output array.

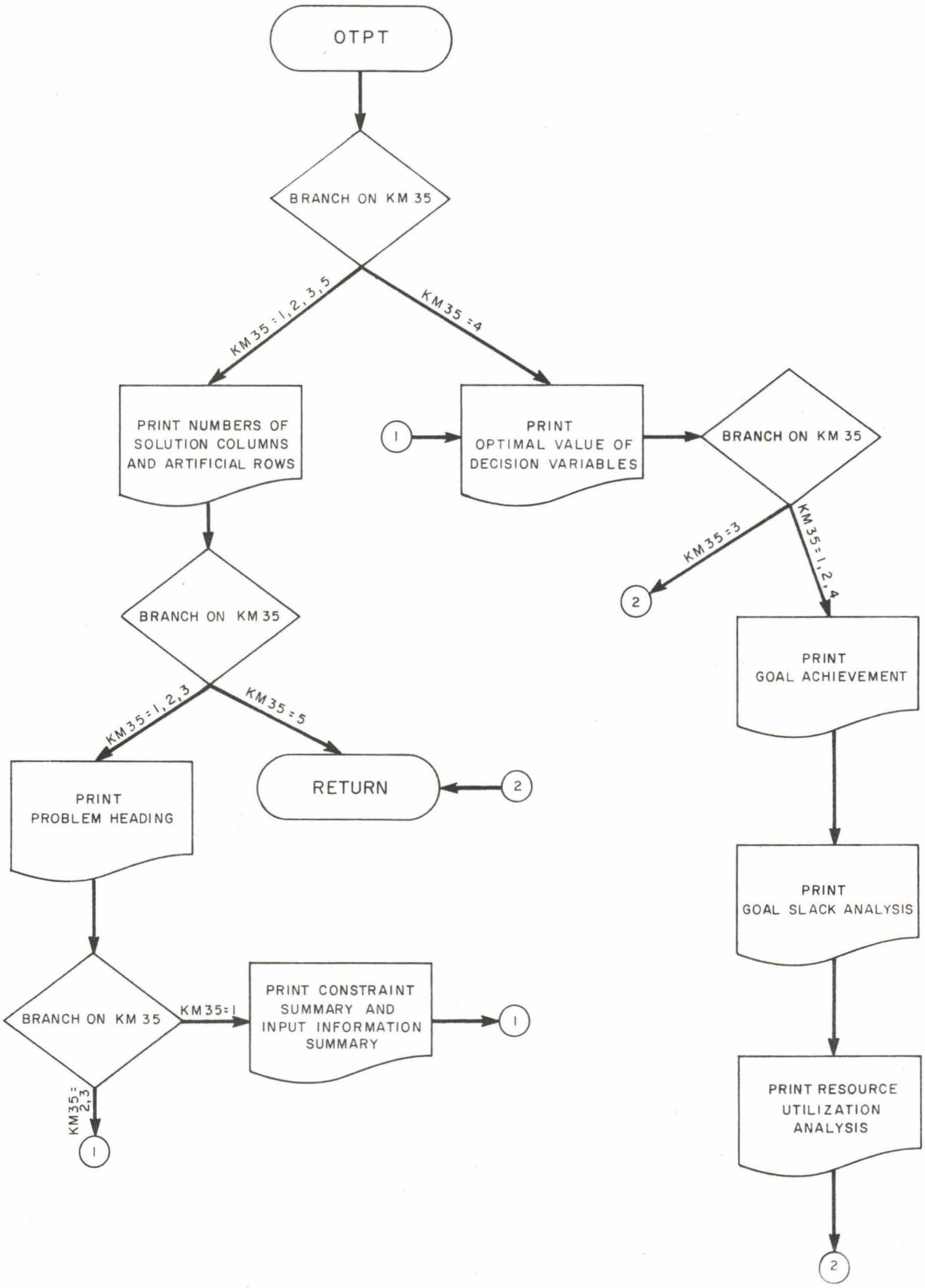

Fig. 9. Flow chart for SUBROUTINE OTPT.

IKJ(J) being non-zero indicates that column J is in the basis. JH(J) being zero indicates an artificial in solution in row J.

```
106 IF(JJ.EQ.0) GO TO 107
    WRITE(WRITEU,140) (KBCD(II),II=1,JJ)
140 FORMAT(1H ,3X,11I4)
```

When the DO loop has reached its upper parameter, a check is made on JJ to see if any numbers have been entered in KBCD; if JJ is zero, none has, and the WRITE statement is skipped.

```
    IF(JJJ.GE.MTB) GO TO 107
    JJ=0
    MTA=JJJ+1
    GO TO 102
```

After the 11 words of KBCD have been written out, JJJ is checked to see if the upper parameter of the DO loop has been reached. If it has not been reached, JJ is set to zero so that 11 more elements may be counted, and MTA is reset so that the DO loop will begin its search at the point it left off when the last WRITE statement was required.

```
107 IF(JIJ.EQ.1) GO TO 108
```

When the upper parameter of the DO loop is reached, this statement is executed to determine if both the rows and columns have been searched. If JIJ is 1 they both have been searched; if zero then a search of the rows for artificials is initiated.

```
    JIJ=1
    WRITE(WRITEU,141)
141 FORMAT(1H0,59HTHESE ARE NUMBERS OF THE ROWS WITH ARTIFICIALS IN SOL
   1UTION.)
    MTB=M
    GO TO 101
```

JIJ is set to 1 indicating that row will be checked for artificials in solution. A heading is written to separate the rows from the columns. MTB is the upper parameter of the DO loop and is set to M, the number of rows. The GO TO sends execution back to the DO loop.

```
108 KM35=KM(35)
    GO TO (109,109,109,112,138),KM35
```

KM(35) can take on values 1-5, these being used in the GO TO statements of OTPT where KM35 is the index which controls the type of output to be generated.

* KM35	Heading	Constraint Summary	Summary of Input Info	Optimal Value of Dec. Var.	Goal Achieve-ment	Goal Slack Analysis	Resource Utilization Analysis
1	X	X	X	X	X	X	X
2	X	X		X	X	X	X
3	X			X			
4				X	X	X	X
5							

* All options print out the solution columns and artificial rows.

```
109 WRITE(WRITEU,174)
174 FORMAT(5(/))
```

Any time a WRITE with this FORMAT number is encountered, six lines are skipped.

```
    WRITE(WRITEU,142)
142 FORMAT(1H0,71(1H$))
    WRITE(WRITEU,149)
    WRITE(WRITEU,143)
143 FORMAT(3H $$,67X,2H$$)
    WRITE(WRITEU,143)
    WRITE(WRITEU,144)
144 FORMAT(3H $$,21X,25HGOAL PROGRAMMING SOLUTION,21X,2H$$)
    WRITE(WRITEU,143)
    WRITE(WRITEU,145)
145 FORMAT(3H $$,23X,21H-PROBLEM DESCRIPTION-,23X,2H$$)
    WRITE(WRITEU,146)(PRONAM(I),I=1,10)
146 FORMAT(3H $$,13X,10A4,14X,2H$$)
    WRITE(WRITEU,143)
    WRITE(WRITEU,147)
147 FORMAT(3H $$,30X,6H-DATE-,31X,2H$$)
    WRITE(WRITEU,148) KBCD(29),KBCD(30)
148 FORMAT(3H $$,29X,2A4,30X,2H$$)
    WRITE(WRITEU,143)
    WRITE(WRITEU,143)
    WRITE(WRITEU,149)
149 FORMAT(1H ,71(1H$))
    WRITE(WRITEU,149)
    GO TO (110,112,112,112),KM35
```

These instructions are simple literal writes which form the heading which is printed at the beginning of each problem.

```
              110 WRITE(WRITEU,150)
              150 FORMAT(1H0,30X,18HCONSTRAINT SUMMARY)
                  WRITE(WRITEU,151)
              151 FORMAT(1H0,3HROW,2X,10HROW RIGHT-,9X,3HROW,8X,3HROW,6X,8HNEGATIV
                 1E,8X,8HPOSITIVE/1H ,5X,9HHAND-SIDE,29X,10HDEVIATIONS,6X,10HDEVIA
                 2TIONS/1H ,3HNO.,4X,5HVALUE,8X,11HDESCRIPTION,4X,4HTYPE,1X,8HPRIO
                 3RITY,1X,6HWEIGHT,1X,8HPRIORITY,1X,6HWEIGHT/1H )
                  DO 111 I=1,M
                  KGI=KG(I)
                  MGI=MG(I)
     (i)          IF(KG(I).EQ.999) KGI=0
                  IF(MG(I).EQ.999) MGI=0
     (ii)         MTA=(I*5)-4
                  MTB=I*5-1
     (iii)        KK=IZIP(I)
                  WRITE(WRITEU,152)NR(I),B(I),(LABL(II),II=MTA,MTB),INAME(KK),
                 1KGI,WW(I),MGI,DY(I)
              152 FORMAT(1H ,A4,F12.2,2X,4A4,2X,A1,5X,I3,4X,F6.2,3X,I3,4X,F6.2)
              111 CONTINUE
```

This block of instructions is used to print out the constraint row (whether goal or otherwise) information as contained in the input data. Everything is straightforward except items (i), (ii), and (iii).

(i) Searches for any priority levels coded 999 (which indicates that no priority level exists for that variable) and sets the output value KGI or MGI to zero so that the output will not be jumbled with 999's.

(ii) MTA and MTB are established as the first and last words in any LABL which contains the description of row I.

(iii) The constraint inequalities are E, G, L, or B and are coded 1, 2, 3, or 4, respectively. To make the output clearer, the letter symbol for the inequality is used. KK is the non-subscripted form of IZIP(I).

```
      K1=N-KM(14)-KM(26)
      WRITE(WRITEU,174)
      WRITE(WRITEU,153)M,MA,N,MB,K1,KM(14),KM(26),KM(8),KM(1)
  153 FORMAT(1H ,25X,28HSUMMARY OF INPUT INFORMATION/1H0,15X,25HNUMBER O
     1F CONSTRAINT ROWS,18(1H.),I5/1H ,15X,33HNUMBER OF NON-ZERO MATRIX
     2ENTRIES,10(1H.),I5/1H ,15X,28HNUMBER OF VARIABLES(COLUMNS),15(1H.)
     3,I5/1H ,15X,20HNUMBER OF PRIORITIES,23(1H.),I5/1H ,15X, 28HNUMBER
     4OF DECISION VARIABLES,15(1H.),I5/1H ,15X,43HNUMBER OF POSITIVE DEV
     5IATIONAL VARIABLES...,I5/1H ,15X,43HNUMBER OF NEGATIVE DEVIATIONAL
     6 VARIABLES...,I5/1H ,15X,30HNUMBER OF ARTIFICIAL VARIABLES,13(1H.)
     7,I5/1H ,15X,43HNUMBER OF ITERATIONS TO FIND THE SOLUTION..,I5)
```

This statement is self explanatory as indicated by the hollerith strings in the FORMAT statement.

```
  112 WRITE(WRITEU,174)
      WRITE(WRITEU,154)
  154 FORMAT(1H ,24X,35HOPTIMAL VALUE OF DECISION VARIABLES)
      K2=N-KM(14)-KM(26)
      KQ=1
      DO 115 J=1,K2
      IF(IKJ(J).EQ.0) GO TO 115
      GO TO (113,114),KQ
  113 WRITE(WRITEU,155)
  155 FORMAT(1H0,17X,8HVARIABLE,10X,11HDESCRIPTION,10X,6HAMOUNT)
      KQ=2
  114 MTA=((M+J)*5)-4
      MTB=(M+J)*5
      IJ=IKJ(J)
      WRITE(WRITEU,156) KN(J),(LABL(II),II=MTA,MTB),X(IJ)
  156 FORMAT(1H ,18X,A4,8X,5A4,3X,F12.2)
  115 CONTINUE
      GO TO (116,117),KQ
  116 WRITE(WRITEU,157)
  157 FORMAT(1H0,22X,41H**NO DECISION VARIABLES ARE IN SOLUTION**)
```

These instructions output a single section of analysis entitled, "OPTIMAL VALUE OF DECISION VARIABLES." K2 is the upper parameter of the DO loop which conducts the search for solution columns (variables), and is the number of the last decision variable column. KQ is 1 until a column in solution is found and 2 thereafter. In this manner the output column headings are only printed once, or are never printed if no variables are in solution. The array IKJ has an element for each column, and if the value for any one column is non-zero, that column is in solution.

MTA and MTB are the first and last words of array LABL which contain the description of column J.

```
      117 GO TO (118,118,138,118),KM35
      118 WRITE(WRITEU,174)
          WRITE(WRITEU,158)
      158 FORMAT(1H ,27X,16HGOAL ACHIEVEMENT)
(i)       DO 124 I=1,MB
          T(1)=0.0
          KQ=0
          K1=N-KM(14)-KM(26)+1
(ii)      DO 122 JJ=K1,N
          IF(JG(JJ).NE.I) GO TO 122
          IF(IKJ(JJ).EQ.0) GO TO 122
          IF(KQ.GT.0) GO TO 119
          WRITE(WRITEU,159) I
      159 FORMAT(1H0,5X,11HGOAL LEVEL ,I5,46H IS NOT ACHIEVED IN THE FOLLOWI
         1NG CONSTRAINTS-)
          KQ=99
(iii) 119 KB1=KB(JJ)
          MTA=(KB1*5)-4
          MTB=KB1*5
          J=IKJ(JJ)
(iv)      IF(KN(JJ).EQ.EQLD) GO TO 120
          IF(KN(JJ).EQ.NEGD) GO TO 120
          IF(KN(JJ).EQ.POSD) GO TO 121
          GO TO 122
      120 WRITE(WRITEU,160)  NR(KB1),(LABL(II),II=MTA,MTB)
      160 FORMAT(1H ,13X,1H*,6X,A4,2H, ,5A4,1H,)
          WRITE(WRITEU,161) X(J)
      161 FORMAT(1H ,20X,20HIS UNDERACHIEVED BY ,F12.2,7H UNITS.)
(v)       T(1)=T(1)+X(J)*WW(KB1)
          GO TO 122
      121 WRITE(WRITEU,160)  NR(KB1),(LABL(II),II=MTA,MTB)
          WRITE(WRITEU,162) X(J)
      162 FORMAT(1H ,20X,19HIS OVERACHIEVED BY ,F12.2,7H UNITS.)
(v)       T(1)=T(1)+X(J)*DY(KB1)
      122 CONTINUE
          IF(KQ.EQ.0) GO TO 123
          IF(T(1).LE..001) GO TO 123
          WRITE(WRITEU,163)
      163 FORMAT(1H ,13X,10H* SUMMARY-)
          WRITE(WRITEU,164) I,T(1)
      164 FORMAT(1H ,16X,5HGOAL ,I5,20H IS NOT ACHIEVED BY ,F12.2,12H WGTD U
         1NITS.)
          GO TO 124
      123 WRITE(WRITEU,165) I
      165 FORMAT(1H0,5X,11HGOAL LEVEL ,I5,50H IS*****************************C
         1OMPLETELY ACHIEVED.)
      124 CONTINUE
```

This block of code analyzes only the columns associated with computational variables which are goal related (i.e. negative and positive deviations with priority levels).

(i) This DO loop establishes a search at all priority levels. T(1) will take on the value of the deviation from a particular goal level. KQ is zero if no deviation from a particular goal level is found, and is 99 if one has been found.

(ii) This DO loop does the actual search of the computational variables (K1 being the number of the first column to contain one), checking for the priority level being evaluated (IF(JG(JJ).NE.I)) and for being in solution (IF(IKJ(JJ).EQ.0)).

(iii) KB1 is the number of the original row which required the creation of column JJ; therefore, the description of row KB1 explains what the deviation in column JJ refers to. MTA and MTB are the first and last words of the array LABL which contain the description of row KB1.

(iv) The IF statements check the type of deviation in column JJ to determine if the deviation was above (overachievement) or below (underachievement) the goal level.

(v) The weighted deviation from a priority level is calculated and added to T(1). WW is the differential weight ("cost") charged for each unit of underachievement, and DY has the same relation to overachievement. X(J) is the number of units at which over- or underachievement occurs.

```
              WRITE(WRITEU,174)
              WRITE(WRITEU,166)
          166 FORMAT(1H ,26X,19HGOAL SLACK ANALYSIS)
              KQ=1
    (i)       DO 129 J=K1,N
   (ii)       IF(IKJ(J).EQ.0) GO TO 129
              IF(JG(J).LT.999) GO TO 129
              IF(KN(J).EQ.POSD) GO TO 125
              IF(KN(J).EQ.NEGD) GO TO 125
              GO TO 129
  (iii)   125 KB1=KB(J)
              IF(IZIP(KB1).LT.4) GO TO 129
   (iv)       GO TO (126,127),KQ
          126 WRITE(WRITEU,167)
          167 FORMAT(66H0THIS SECTION ANALYZES GOAL CONSTRAINTS WITH -B- TYPE IN
             1EQUALITIES/70H WHERE EITHER A NEGATIVE OR POSITIVE DEVIATION IS NO
             2T GIVEN A PRIORITY/70H LEVEL. THE VALUE WILL THEN REFLECT THE AMOU
             3NT BY WHICH THE EXACT GOAL/68H WAS NOT ACHIEVED, EVEN THOUGH THE M
             4INIMUM OR MAXIMUM GOAL LEVEL WAS/10H ACHIEVED./1H0,4X,3HROW,12X,4H
             5GOAL,12X,5HEXACT,9X,8HNEGATIVE,6X,8HPOSITIVE/1H ,3X,6HNUMBER,7X,11
             6HDESCRIPTION,6X,10HGOAL LEVEL,8X,5HSLACK,9X,5HSLACK)
              KQ=2
    (v)   127 MTA=KB1*5-4
              MTB=KB1*5
              IJ=IKJ(J)
              IF(KN(J).EQ.NEGD) GO TO 128
              WRITE(WRITEU,168) NR(KB1),(LABL(II),II=MTA,MTB),B(KB1),X(IJ)
          168 FORMAT(1H0,3X,A4,3X,5A4,1X,F12.2,7X,7H0000.00,2X,F12.2)
              GO TO 129
          128 WRITE(WRITEU,169) NR(KB1),(LABL(II),II=MTA,MTB),B(KB1),X(IJ)
          169 FORMAT(1H0,3X,A4,3X,5A4,1X,F12.2,2X,F12.2,7X,7H0000.00)
          129 CONTINUE
   (vi)       GO TO (130,131),KQ
          130 WRITE(WRITEU,170)
          170 FORMAT (64H0**ALL GOALS WITH CONSTRAINT TYPE -B- WERE EITHER UNATT
             1AINED AND/65H TREATED IN THE GOAL ATTAINMENT SECTION, OR THE GOALS
             2 WERE MET AT/65H THEIR SPECIFIED LEVEL. THEREFORE THIS SECTION OF
             3ANALYSIS IS NOT/10H NEEDED.**)
```

This block of code generates the output for analyzing B type constraints where either the negative or positive deviations are not associated with a goal priority level.

(i) The DO loop processes all computational columns (K1 being the number of the first column which contains a computational variable).

(ii) These four IF statements filter out all columns which do not meet the following conditions:

-- the column is in solution,

-- there exists a priority level for that column (999 indicates that no priority level is assigned that column),

-- the column represents either a negative or positive deviation (NEGD or POSD).

(iii) When a column is found which meets the conditions of (ii), KB1 takes on the number of the original row which required the creation of column J. The following IF statement insures that the constraint type is 4 (equivalent of B).

(iv) KQ is 1 until a column is found which meets the criteria of (ii) and (iii), and is 2 thereafter. In this manner it is determined whether or not the general heading of this analysis section should be printed, and if so, insures that it is printed only once.

(v) MTA and MTB are the first and the last words of array LABL which contain the description of row KB1, which is the original row which required the creation of the deviational variable in column J.

(vi) If, at the termination of the DO loop, KQ is still 1 (no deviational variable of the specified type has been encountered) a diagnostic is printed.

```
         131 WRITE(WRITEU,174)
             WRITE(WRITEU,171)
         171 FORMAT(1H ,26X,29HRESOURCE UTILIZATION ANALYSIS)
             KQ=1
   (i)       DO 136 JJ=K1,N
             IF(IKJ(JJ).EQ.0) GO TO 136
             IF(JG(JJ).LT.999) GO TO 136
             IF(KN(JJ).EQ.SLCK) GO TO 132
             IF(KN(JJ).EQ.SRPL) GO TO 132
             GO TO 136
  (ii)   132 GO TO (133,134),KQ
         133 WRITE(WRITEU,172)
         172 FORMAT(1H0,5X,3HROW,10X,8HRESOURCE,11X,5HEXACT,8X,8HRESOURCE,6X,8H
            1RESOURCE/1H ,4X,6HNUMBER,6X,11HDESCRIPTION,6X,14HRESOURCE LEVEL,3X
            2,8HNOT-USED,4X,13HOVER-PRODUCED)
             KQ=2
```

```
(iii)    134 KB1=KB(JJ)
             MTA=KB1*5-4
             MTB=KB1*5
             IJ=IKJ(JJ)
 (iv)        IF(KN(JJ).EQ.SLCK) GO TO 135
             WRITE(WRITEU,168) NR(KB1),(LABL(II),II=MTA,MTB),B(KB1),X(IJ)
             GO TO 136
         135 WRITE(WRITEU,169) NR(KB1),(LABL(II),II=MTA,MTB),B(KB1),X(IJ)
         136 CONTINUE
  (v)        GO TO (137,138),KQ
         137 WRITE(WRITEU,173)
         173 FORMAT(1H0,6X,57H**ALL RESOURCES, AS EXPRESSED IN CONSTRAINTS, WER
            1E USED**)
```

This block of instructions analyzes those computational variables which were required by non-goal constraints.

(i) This DO loop processes all computational variable columns where K1 is the number of the first column to contain a computational variable. The four IF statements which follow insure that the columns (variables) comply with the following criteria:

-- the column is in solution (IF(IKJ(JJ).EQ.0));

-- the column is not associated with a goal priority level (IF(JG(JJ). LT.999)), where 999 indicates that no priority level exists;

-- and that the variable is a slack or a surplus (SLCK or SRPL).

(ii) KQ is 1 if no slack or surplus variables are found and 2 thereafter. In this manner it is determined whether or not the output heading should be printed, and insures that it is only printed once.

(iii) KB1 is the number of the original row which required the creation of column JJ. MTA and MTB are the first and last words of array LABL which contain the description of row KB1.

(iv) This IF statement directs the printing of the value of the variable in solution under the proper column heading. A SLCK will cause the value of the variable to be placed under the "NOT-USED" column while the failure of the IF (SRPL implied) will position the value under "OVER-PRODUCED."

(v) If, at the termination of the DO loop, KQ is still 1 (indicating that no slacks or surplus variables have been found), a diagnostic is printed.

```
138 WRITE(WRITEU,174)
    KM(35)=1
    RETURN
```

This WRITE statement is executed at the end of each use of OTPT. KM(35) is also set to 1 indicating that a full output is required unless otherwise specified.

SUBROUTINE PIV (IR, COMMON PARAMETERS) performs the task of creating pivot vectors (eta vectors or transformation vectors are synonymous). The column which will be pivoted into solution has already been selected, updated to its current value by subroutine JMY, and stored in array Y. The flow chart for SUBROUTINE PIV is shown in Fig. 10.

Unique formal parameters:

IR - is the number of the row for which the column in array Y is being pivoted.

```
KM(15)=KM(15)+1
```

KM(15) is the counter for the number of pivots performed. At present this information is not printed out anywhere.

```
T(1)=Y(IR)
T(3)=X(IR)/T(1)
T(4)=ABS(Z(5)*T(1))
X(IR)=0.
Y(IR)=-1.
K=LE(ME+1)
```

T(1) is set to the value of the pivot element. T(3) is the ratio of the right-hand-side of the pivot row to the pivot element. This will be used to update all of the right-hand-side values. T(4) is the tolerance above which each element of the array Y must conform before an eta entry can be made. This is required to insure that the division in creating the eta entries is meaningful. The procedure for creating an eta vector is as follows:

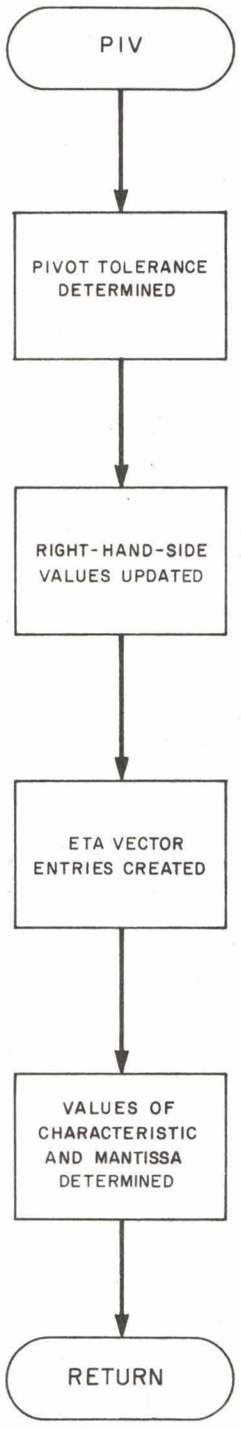
PIV
PIVOT TOLERANCE DETERMINED
RIGHT-HAND-SIDE VALUES UPDATED
ETA VECTOR ENTRIES CREATED
VALUES OF CHARACTERISTIC AND MANTISSA DETERMINED
RETURN

Fig. 10. Flow chart for SUBROUTINE PIV.

$$\begin{array}{ll} \left[\begin{array}{c} a_{1,t} \\ a_{2,t} \\ \cdot \\ \cdot \\ \cdot \\ \boxed{a_{IR,t}} \\ \cdot \\ \cdot \\ \cdot \\ a_{M,t} \end{array}\right] & \begin{array}{l} \\ \\ \\ \\ \\ \text{Element of the pivot column cor-} \\ \text{responding to the pivot row.} \\ \\ \\ \\ \end{array} \end{array}$$

where t is the number of the pivot column. The eta vector is calculated from the current value of the pivot column in the following manner:

$$\begin{bmatrix} -a_{1,t}/a_{IR,t} \\ -a_{2,t}/a_{IR,t} \\ \cdot \\ \cdot \\ \cdot \\ 1/a_{IR,t} \\ \cdot \\ \cdot \\ \cdot \\ -a_{M,t}/a_{IR,t} \end{bmatrix}$$

Thus by requiring that element $a_{i,t}$ be at least Z(5)*T(1), it can be seen that a meaninglessly small number will not be formed.

X(IR) is set to zero so that the DO loop which will recalculate the current values of X(I) will not make X(IR) larger than it should be (this is because T(3) is the new current value of X(IR)). Y(IR) is set to -1 so that the DO loop which calculates the other eta entries can calculate the eta entry corresponding to the pivot row. K is the number of the next eta entry to be formed.

```
              DO 104 I=1,M
   (i)        IF (ABS(Y(I))-T(4)) 104,104,101
  (ii)    101 X(I)=X(I)-T(3)*Y(I)
 (iii)        IF (K-KM(21)+M) 103,103,102
          102 KM(13)=1
  (iv)    103 E(K)=-Y(I)/T(1)
   (v)        IE(K)=I
  (vi)        K=K+1
          104 CONTINUE
```

This block of code checks each element in the pivot column (Y) to see if an eta entry can be made, and also updates the right-hand-side of each row.

(i) The tolerance T(4) is applied to element I of array Y. If the result is less than or equal to zero, no eta entry will be made.

(ii) The right-hand-side of each row is updated.

(iii) A check is made to insure that the number of eta entries created does not exceed the storage allowed for those entries.

(iv) E(K) is the value of eta entry number K and is created in the manner described in the last block of code.

(v) IE(K) is the number of the row which created eta entry number K.

(vi) K is incremented to the number of the next eta entry to be created.

```
    Y(IR)=T(1)
    KM(10)=K-1
    ME=ME+1
```

Y(IR) is reset to its original value so that the value of a current column can be printed out (if desired) in its correct form. KM(10) is the actual number of eta entries. ME is the number of pivot vectors formed.

```
        Z(7)=Z(7)*T(1)
    105 IF (ABS(Z(7))-1.0) 106,109,107
    106 Z(7)=Z(7)*10.
        KM(18)=KM(18)-1
        GO TO 105
    107 IF (ABS(Z(7))-10.0) 109,108,108
    108 Z(7)=Z(7)/10.
        KM(18)=KM(18)+1
        GO TO 107
```

These instructions calculate the value of the characteristic (KM(18)) of the logarithm of the absolute value of the pivot element and the antilogarithm (Z(7)) of the mantissa.

```
109 IF (ME-KM(20)) 110,110,111
110 IP(ME)=IR
    LE(ME+1)=K
```

The IF statement checks to see if the number of eta vectors allowed has been exceeded. If not, IP, the number of the pivot row which created eta vector ME, is set to IR, and LE(ME+1) establishes the number of the eta entry which will begin the next eta vector.

```
    IF(KM(31).LE.0) GO TO 112
    WRITE(WRITEU,113) KM(10)
113 FORMAT(1H ,10HTHERE ARE ,I5,18H ETA FILE ENTRIES.)
    WRITE(WRITEU,114) KM(1),(X(I),I=1,M)
114 FORMAT(1H ,19HR-H-S AT ITERATION ,I3, 4(2X,F9.1))
    GO TO 112
```

If KM(31) is greater than zero, the option to print out the current value of the right-hand-side is performed.

```
111 KM(13)=1
112 RETURN
```

KM(13) is 1 when either the number of allowed eta entries or eta vectors has been exceeded.

SUBROUTINE PRI (KLM, COMMON PARAMETERS) performs the task of incrementing the number of the priority level being operated on when the deviations from the existing priority have been reduced to the minimum possible points. Also, if no pivot column can be found when artificials are in solution, an infeasible condition is printed. The flow chart for SUBROUTINE PRI is shown in Fig. 11.

Unique formal parameters:

KLM - the number of the pivot column. If KLM is ever returned as zero, either the problem is infeasible (KM(35) will be 5) or an optimal solution has been found.

```
    KLM=0
    IREJ=0
```

KLM is initialized at zero, indicating that at present no pivot column has been chosen. IREJ is set at zero so that when BRO is called, a complete objective function at priority level KM(16) will be formed.

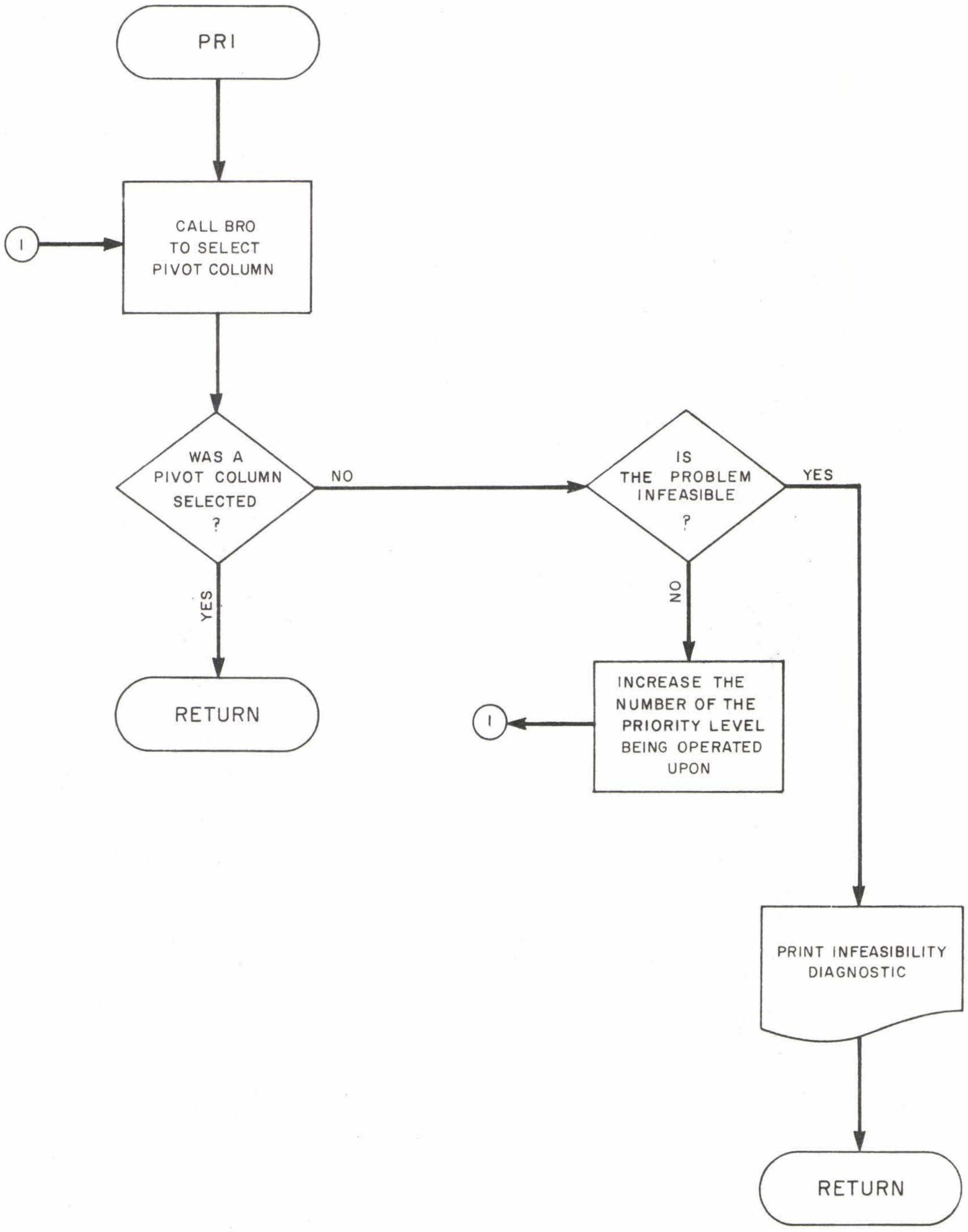
PRI
CALL BRO TO SELECT PIVOT COLUMN
1
WAS A PIVOT COLUMN SELECTED ?
NO
YES
RETURN
IS THE PROBLEM INFEASIBLE ?
YES
NO
INCREASE THE NUMBER OF THE PRIORITY LEVEL BEING OPERATED UPON
1
PRINT INFEASIBILITY DIAGNOSTIC
RETURN

Fig. 11. Flow chart for SUBROUTINE PRI.

```
101 CALL BRO(NK,IREJ,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,
   1X,Y,KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
    KLM=NK
    IF(NK.EQ.0) GO TO 102
    GO TO 105
```

BRO is called to create an objective function, and to identify the column which contains the highest positive coefficient. The number of the column is returned to PRI in NK and is equated with KLM.

```
102 IF(KM(16).EQ.0) GO TO 103
    IF(KM(16).EQ.MB) GO TO 104
    KM(16)=KM(16)+1
    GO TO 101
```

If NK is returned from BRO as zero these instructions are executed. If KM(16) is not zero, then the existing priority has been minimized to the greatest possible extent; therefore, KM(16) is increased by 1, and BRO is called again.

```
103 WRITE(WRITEU,106)
106 FORMAT(1H0,60HAN -E- OR -G- TYPE CONSTRAINT MAKES THIS PROBLEM INF
   1EASIBLE.)
    KM(35)=5
```

When an infeasible situation is found, the above diagnostic is printed and KM(35) is set to 5 so that the solution columns and artificial rows will be printed out when OTPT is called.

```
104 KLM=0
105 RETURN
```

If the number of the pivot column, KLM, is ever returned to CNTRL as zero, these instructions are executed.

SUBROUTINE REIN (ITYPE, COMMON PARAMETERS) performs two major tasks; reinversion and feasible crashing. Reinversion is that process whereby the eta file is recreated to represent the inverse matrix in its most concise form (the eta file is the product form of the inverse matrix). Thus the most concise eta file will consist of a single pivot vector (eta vector) for each column in solution. The flow chart for SUBROUTINE REIN is shown in Fig. 12.

Feasible crashing has the purpose of driving all possible artificial variables from solution.

Unique formal parameters:

ITYPE - controls the WRITE statements which print the purpose for which REIN was called.

```
      IF(KM(36).EQ.0) GO TO 155
```

If KM(36) is zero, the computational variables for initial solution have not yet been created by ART, and a diagnostic will be printed.

```
      KM(19)=KM(1)
      KK=0
      KBAD=0
      KM(13)=0
      KM(17)=0
```

Control variables are set. KM(19) is the number of the last iteration before reinversion or crashing. KK is the number of pivots performed by REIN. KBAD is the number of columns which will have no pivot row. KM(13) is 1 when reinversion is needed. KM(17) is the number of iterations since the last reinversion.

```
  (i)       IF(KM(33))101,104,101
 (ii)   101 DO 103 J=1,N
            IF(IKJ(J))102,102,103
        102 IKJ(J)=-12345
        103 CONTINUE
(iii)       IF(LE(1))112,104,112
```

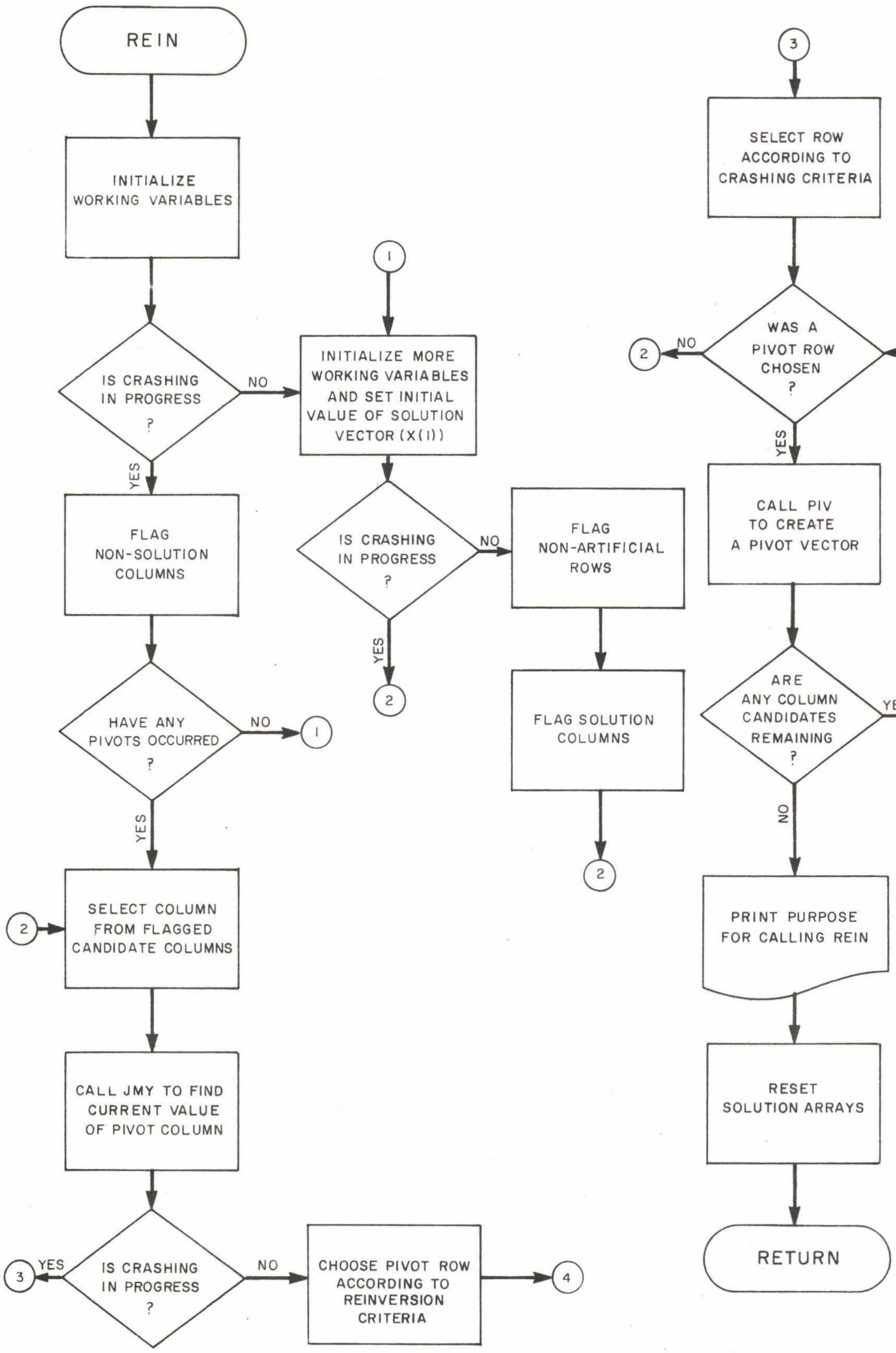
REIN
INITIALIZE WORKING VARIABLES
IS CRASHING IN PROGRESS ?
NO
YES
FLAG NON-SOLUTION COLUMNS
HAVE ANY PIVOTS OCCURRED ?
NO
1
YES
2
SELECT COLUMN FROM FLAGGED CANDIDATE COLUMNS
CALL JMY TO FIND CURRENT VALUE OF PIVOT COLUMN
3
YES
IS CRASHING IN PROGRESS ?
NO
CHOOSE PIVOT ROW ACCORDING TO REINVERSION CRITERIA
4
1
INITIALIZE MORE WORKING VARIABLES AND SET INITIAL VALUE OF SOLUTION VECTOR (X(I))
IS CRASHING IN PROGRESS ?
YES
2
NO
FLAG NON-ARTIFICIAL ROWS
FLAG SOLUTION COLUMNS
2
3
SELECT ROW ACCORDING TO CRASHING CRITERIA
2
NO
WAS A PIVOT ROW CHOSEN ?
4
YES
CALL PIV TO CREATE A PIVOT VECTOR
ARE ANY COLUMN CANDIDATES REMAINING ?
YES
2
NO
PRINT PURPOSE FOR CALLING REIN
RESET SOLUTION ARRAYS
RETURN

Fig. 12. Flow chart for SUBROUTINE REIN.

(i) This statement insures that the following instructions are executed only if feasible crashing is to be done (KM(33) is non-zero for crashing).

(ii) The DO loop is indexed to the number of columns. Each column is checked for not being in solution (IKJ(J) will be zero or negative), and these columns are flagged as being candidates for pivoting out artificials. The code -12345 identifies this condition.

(iii) If LE(1) is zero, there have been no pivots made; therefore, certain variables are initialized. If LE(1) is greater than zero, then the crashing process will begin.

```
(i)     104 ME=0
            KM(32)=M
            KM(10)=0
            KM(18)=0
            Z(7)=1.0
            LE(1)=1
(ii)        DO 105 I=1,M
        105 X(I)=B(I)
```

(i) These instructions are executed if reinversion is to occur or if crashing is to be done before any pivoting takes place. KM(32), the number of artificials in solution, is set to the number of rows. The number of existing eta vectors (ME), and the number of eta entries (KM(10)) are set to zero. The characteristic (KM(18)) and mantissa antilog (Z(7)) are set to starting values. LE(1), the number of the first eta entry in eta vector 1, is set to 1.

(ii) The current solution vector, X(I), is initialized at the original right-hand-side value, B(I).

```
(i)         IF(KM(33))112,106,112
(ii)    106 DO 108 I=1,M
            IF(JH(I))107,108,107
        107 JH(I)=12345
        108 CONTINUE
(iii)       DO 111 J=1,N
            IF(IKJ(J))109,111,110
        109 IKJ(J)=0
            GO TO 111
        110 IKJ(J)=-12345
        111 CONTINUE
```

(i) The IF statement allows execution of the instructions which follow only if reinversion is taking place (KM(33) is zero for reinversion).

(ii) The DO loop checks rows for columns in solution (JH(I) is non-zero if a column is in solution in row I). Those which have a column in solution are flagged with 12345. This flagging process occurs so that when row selection is being made, no pivots will be made into rows which contain artificials (JH(I)=0) and so that the same row is not pivoted on twice.

(iii) This DO loop inspects the array IKJ for columns which are in solution. It is intended that all non-solution columns are marked by a value of zero, and all solution columns are marked by a value of -12345. In this manner, when reinverting or crashing, candidates for entry into solution are the solution columns (-12345).

```
  (i)    112 IF(KM(32))113,128,113
 (ii)    113 KM(23)=12345
             DO 116 JJ=1,N
             IF(IKJ(JJ))114,116,116
         114 KM(24)=KL(JJ+1)-KL(JJ)
             IF(KM(24)-KM(23))115,116,116
         115 KM(23)=KM(24)
             J=JJ
         116 CONTINUE
(iii)        IF(KM(23)-12345)117,128,117
 (iv)    117 IKJ(J)=0
             KM(6)=KN(J)
             CALL JMY(J,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,X,Y,
            1KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
```

(i) If KM(32) is zero, then no more eligible pivot rows exist and execution goes to the termination instructions.

(ii) Of all the columns which have been designated as pivot candidates (IKJ(JJ)=12345), the column with the least number of non-zero entries (original data) is selected as the pivot column.

(iii) This statement checks to see if any column was selected. If not, execution goes to the termination instructions.

(iv) IKJ(J) is set to zero (as opposed to -12345) so that this column will not be selected again. KM(6) is the name of the column selected. JMY is called to calculate the current value of column J.

```
   (i)          IR=0
  (ii)          IF(KM(33))140,118,140
 (iii)      118 T(2)=Z(1)
                DO 121 I=1,M
                IF(JH(I)-12345)121,119,121
            119 IF(ABS(Y(I))-T(2))121,121,120
            120 T(2)=ABS(Y(I))
                IR=I
            121 CONTINUE
```

(i) IR is set to zero before any attempt is made to select a pivot row.

(ii) This statement insures that the instructions which follow are executed only if reinversion is taking place.

(iii) The pivot row during reinversion is determined by the following criteria:

-- the pivot element in column J is not zero,

-- the pivot element has the highest absolute value.

```
            122 IF(IR)124,123,124
```

When the number of the pivot row has been determined (IR), whether chosen for reinversion or crashing, it needs to be determined if that row number is zero.

```
            123 KBAD=KBAD+1
                GO TO 112
```

If IR is zero (indicating that no pivot row was found for column J), then KBAD is increased by 1. KBAD indicates the number of column candidates for which no pivot row was found. This number has little significance to the user and is not printed out.

```
   (i)      124 JH(IR)=J
                IKJ(J)=IR
                KM(32)=KM(32)-1
  (ii)          IF(KM(31).LE.0) GO TO 125
                WRITE(WRITEU,157) J
            157 FORMAT(1H0,16HPIVOT COLUMN IS ,I5)
                WRITE(WRITEU,158) IR
            158 FORMAT(1H ,13HPIVOT ROW IS ,I5)
 (iii)      125 IF(KM(23)-1)127,126,127
  (iv)      126 JJ=KL(J)
                IF(A(JJ)-1.0)127,112,127
   (v)      127 CALL PIV(IR,KG,WW,JG,IZIP,MG,DY,A,IA,KL,KN,NR,B,JH,KB,X,Y,
               1KNT,LABL,W,CAX,IKJ,E,IE,IP,LE)
                KK=KK+1
                IF(KM(13))128,112,128
```

(i) When a pivot column and row are found, the variable JH(IR) is set to J indicating that column number J is in solution in row IR, and IKJ(J) is set to IR indicating that IR is the row in which J is in solution. JH is row oriented while IKJ is column oriented. The number of eligible pivot rows (KM(32)) is decreased by one.

(ii) If KM(31) is non-zero, the option to write out the number of the pivot column and row is executed.

(iii) This statement determines whether or not a single non-zero coefficient exists in column J.

(iv) If a single non-zero coefficient exists in column J and the value of that coefficient is 1, then no pivot vector is formed and execution is sent to find another pivot column.

(v) If the above described condition does not hold, PIV is called to create a pivot vector, KK is increased by 1 to count the number of pivot vectors created by the call to REIN, and KM(13) is checked for being non-zero, which would indicate that the storage for eta vectors or eta entries has been exceeded.

```
128 DO 130 I=1,M
    IF(JH(I)-12345)130,129,130
129 JH(I)=0
130 CONTINUE
    DO 132 J=1,N
    IF(IKJ(J))131,132,132
131 IKJ(J)=0
132 CONTINUE
```

These instructions are executed only when the reinversion or crashing process is complete. Any artificial row which was flagged 12345 for reinversion is reset to zero, and any solution column which was coded -12345 and not pivoted on is reset to zero.

```
    GO TO (133,134,135,136,137),ITYPE
133 WRITE(WRITEU,159)KM(1)
159 FORMAT(1H0,41HREINVERSION BECAUSE OF TIME AT ITERATION ,I5)
    GO TO 138
134 WRITE(WRITEU,160) KM(1)
160 FORMAT(1H0,42HREINVERSION BECAUSE OF SPACE AT ITERATION ,I5)
    GO TO 138
135 WRITE(WRITEU,161) KM(1)
161 FORMAT(1H0,45HREINVERSION AT OPTIMAL SOLUTION AT ITERATION ,I5)
    GO TO 138
```

```
      136 WRITE(WRITEU,162)KK
      162 FORMAT(1H0,14HCRASHING WITH ,I5,8H PIVOTS.)
          GO TO 138
      137 WRITE(WRITEU,163) KK
      163 FORMAT(1H0,21HADVANCED BASIS START.,I5,20H COLUMNS PIVOTED IN.)
```

ITYPE controls which WRITE statement is executed.

```
      138 KM(33)=0
          IF(KM(13))139,156,139
```

Before control is returned to CNTRL, KM(33) is set to zero, indicating that reinversion is the default option of REIN, and KM(13) is checked for being non-zero, indicating that eta entry storage has been exceeded.

```
      139 WRITE(WRITEU,164)
      164 FORMAT(26H TOO MANY ETA FILE ENTRIES)
          KM(10)=LE(ME)-1
          ME=ME-1
          STOP 20
```

If eta entry storage has been exceeded these instructions are executed.

```
  (i)     140 AA=1.0E+20
              CC=0.0
              DO 145 I=1,M
              IF(JH(I))141,145,141
          141 IF(Y(I)-Z(1))145,145,142
          142 IF(X(I)+Z(2))145,143,143
          143 IF(X(I)-Z(2))154,154,144
          144 AA=AMIN1(AA,X(I)/Y(I))
          145 CONTINUE
  (ii)    146 KKK=0
              DO 153 I=1,M
              IF(JH(I))153,147,153
          147 KKK=1
              IF(ABS(Y(I))-Z(1))153,153,148
          148 IF(ABS(X(I))-Z(2))151,151,149
          149 BB=X(I)/Y(I)
              IF(BB)153,153,150
          150 IF(BB-AA)151,151,153
          151 IF(ABS(Y(I))-CC)153,153,152
          152 CC=ABS(Y(I))
              IR=I
          153 CONTINUE
              IF(KKK.EQ.0) GO TO 128
              GO TO 122
  (iii)   154 AA=0.0
              GO TO 146
```

This block of code is used to select a pivot row when feasible crashing is in progress.

(i) These instructions identify the smallest positive ratio (right-hand-side value/corresponding pivot column element). This value is stored in AA. If a negative right-hand-side is found, the DO loop is left and (iii) is executed leaving AA equal to zero.

(ii) These instructions seek to find a pivot row which meets the following criteria:

-- the row contains an artificial variable;

-- the ratio of the right-hand-side value to the corresponding pivot column element must yield a positive result (i.e. the ratio elements must be both positive or both negative);

-- this ratio is stored in BB;

-- the ratio BB must be less than or equal to AA;

-- of those rows which must meet the above three criteria, the one with the pivot column element of highest absolute value is chosen.

If KKK is zero at completion of this DO loop, this indicates that there are no artificial rows and execution is sent to the termination instructions. If KKK is 1 then execution is sent to evaluate the pivot row chosen and if certain criteria are met, a pivot is performed.

```
155 WRITE(WRITEU,165)
165 FORMAT(1H0,71HEXECUTION CANNOT PROCEED UNTIL THE INITIAL FEASIBLE
   1SOLUTION IS SET UP./1H ,35HA SNGL OR SOLV COMMAND IS REQUIRED.)
156 RETURN
```

This diagnostic is written when REIN is entered before the variables needed for initial solution have been created by ART. KM(36) will be zero in this situation.

SUBROUTINE RHS (K, COMMON PARAMETERS) performs the task of entering the data pertaining to rows. The flow chart for SUBROUTINE RHS is shown in Fig. 13.

Unique formal parameters:

K - when data is found in columns 1-4 of the RHS data cards, it is stored in K and returned to INPT. This data should be a command word.

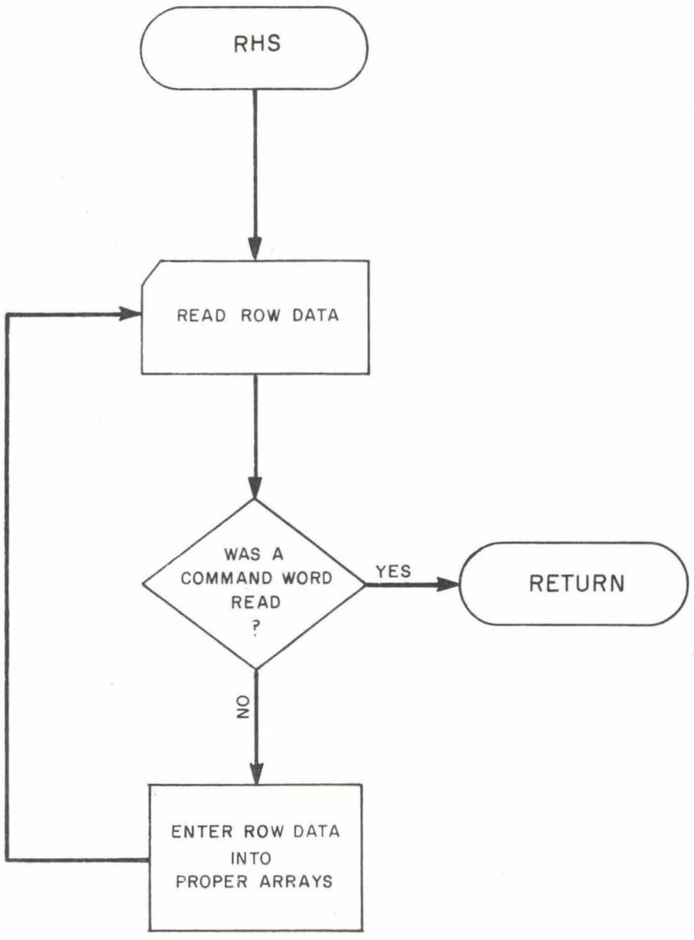

Fig. 13. Flow chart for SUBROUTINE RHS.

```
          ERFLG=0
          DO 101 I=1,4
      101 NVALS(I)=0
```

The error flag is initialized to zero indicating no errors for input data. The DO loop initializes all elements of the array NVALS to zero. The array is used to store counters for the number representing each type of inequality on the RHS input cards.

```
          IF(KM(37).LE.0) GO TO 102
          WRITE(WRITEU,117)
      117 FORMAT(1H0,35X,8HNEGATIVE,8X,8HPOSITIVE/1H ,7X,52HROW  RIGHT-HAND-
         1 ROW      DEVIATIONS       DEVIATIONS/1H ,7X,54HNAME  SIDE VALUE  T
         2YPE PRIORITY WEIGHT PRIORITY WEIGHT/1H )
```

If KM(37) is non-zero, the option to echo the input data is performed, and the heading for the data is printed.

```
      102 READ(READU,118)K,KBCD(3),JJ,AA,IT,IG,PW,LG,QW,(KBCD(II),II=4,8)
      118 FORMAT(3A4,F12.0,A1,2(I5,F12.0),1X,5A4)
```

This READ statement reads all data for this subroutine.

Cols.	1-4	= K - any command words
Cols.	5-8	= KBCD(3) - blank
Cols.	9-12	= JJ - row name or symbol
Cols.	13-24	= AA - value of right-hand-side
Col.	25	= IT - symbol for type of inequality
Cols.	26-30	= IG - priority level for negative deviational variables
Cols.	31-42	= PW - differential weight for negative deviational variables
Cols.	43-47	= LG - priority level for positive deviational variables
Cols.	48-59	= QW - differential weight for positive deviational variables
Col.	60	- leave blank
Cols.	61-80	= KBCD(4-8) - description of row.

```
          IF(K-KM(25)) 104,103,104
```

The input variable K is checked for being non-blank, which would indicate that a command word has been reached.

```
      103 IF(KM(37).LE.0) GO TO 107
          WRITE(WRITEU,119) JJ,AA,IT,IG,PW,LG,QW
      119 FORMAT(1H ,7X,A4,1X,F12.2,2X,A1,6X,I3,3X,F6.2,4X,I3,3X,F6.2)
          GO TO 107
```

If KM(37) is non-zero, the option to echo the input data is executed.

```
  (i)    104 IF(M.GT.NA) GO TO 114
 (ii)    105 NSUM=0
             DO 106 I=1,4
             NSUM=NSUM+NVALS(I)
         106 CONTINUE
             IF(NSUM-NA) 115,116,115
```

This block checks for errors in the right-hand-side data cards.

(i) This IF statement checks the number of rows created for possibly exceeding the number of rows stated on the parameter card.

(ii) Summation for the number of different types of inequalities to check against the total number of rows read by the program.

```
         107 M=M+1
```

If no command word is read (K is blank), then the number of rows is increased by one.

```
             DO 108 I=1,4
             IF (IT.EQ.INAME(I)) NVALS(I)=NVALS(I)+1
         108 CONTINUE
```

This DO loop does the counting for the numbers of each type of inequality in each row. Array NVALS has the elements arranged in the same order as the elements in the array INAME.

```
             DO 109 J=1,M
             IF(NR(J)-JJ)109,113,109
         109 CONTINUE
```

This DO loop compares the name of the row just read (JJ) with the names of all other rows created to this point. If a row name is duplicated, execution is sent to an error diagnostic.

```
             IF(IG.EQ.0) IG=999
             IF(LG.EQ.0) LG=999
```

If there is no priority level associated with a negative or positive deviation, then the input values of IG or LG will be zero. For internal manipulation, this condition is to be coded 999.

```
          KG(M)=IG
          WW(M)=PW
          MG(M)=LG
          DY(M)=QW
          NR(M)=JJ
          B(M)=AA
```

All of the card image row attributes are transferred to row oriented arrays.

```
          M5=M*5-4
          DO 110 II=4,8
          LABL(M5)=KBCD(II)
          M5=M5+1
      110 CONTINUE
```

The row description which was input into KBCD(4-8) is transferred to array LABL. Five words of LABL are allocated for each row description (four characters per word). Therefore, M (the number of the row) times 5 will be the last word of LABL allocated to row M, and (M*5)-4 will be the first word of LABL allocated to row M.

```
          DO 111 I=1,4
          II=I
          IF(IT-INAME(I))111,112,111
      111 CONTINUE
          WRITE(WRITEU,120)  JJ
      120 FORMAT(1H0,30HRHS DATA CARD, WITH ROW NAMED ,A4/
         11H ,46HHAS AN INVALID INEQUALITY SYMBOL IN COLUMN 25.)
          STOP
      112 IZIP(M)=II
          GO TO 102
```

This DO loop checks the type of inequality symbol input on each RHS card. Array INAME contains the possible symbols for the types of inequalities. If a search of INAME does not yield a match, the WRITE statement prints an error diagnostic. If a search of INAME does yield a match, a numerical value is stored in the array IZIP representing the symbol for use in computed GO TO statements.

```
113 WRITE(WRITEU,121)
121 FORMAT(1H ,39HTWO ROWS HAVE BEEN GIVEN THE SAME NAME.)
    ERFLG=1
    GO TO 102
114 WRITE(WRITEU,122)
122 FORMAT(1H0,41HMAXIMUM NUMBER OF ROWS HAS BEEN EXCEEDED.)
    ERFLG=1
    GO TO 105
115 WRITE (WRITEU,123)
    FORMAT (67H0THE NUMBER OF ROWS WITH EACH TYPE OF INEQUALITY DOES N
   1OT MATCH THE/67HPARAMETER CARD VALUES, PLEASE CHECK THE RHS CARDS
   2AND THE PARAMETER/6H CARD.)
    ERFLG=1
116 RETURN
```

These are self explanatory error diagnostics.

SUBROUTINE ROW (IR, KLM, COMMON PARAMETERS) performs the task of determining if priority levels which have already been minimized will be violated by a pivot on column number KLM. If no violations occur, a pivot row is selected. The flow chart for SUBROUTINE ROW is shown in Fig. 14.

Unique formal parameters:

IR - when a pivot row is selected, the number of that row is returned to CNTRL through this argument.

KLM - This is the number of the pivot column, and is passed to ROW through this argument.

```
IR=0
IF(KM(30).EQ.1) GO TO 107
IF(KM(16).EQ.1) GO TO 107
```

IR is initially set to zero, indicating that no pivot row has been selected. If the problem at this stage is infeasible (KM(30) is 1) or the first priority level is being operated on (KM(16) is 1), then the section of code which checks for violation of previously minimized priority levels is bypassed.

```
K2=KM(16)-1
```

K2 is the number of priority levels which have thus far been minimized.

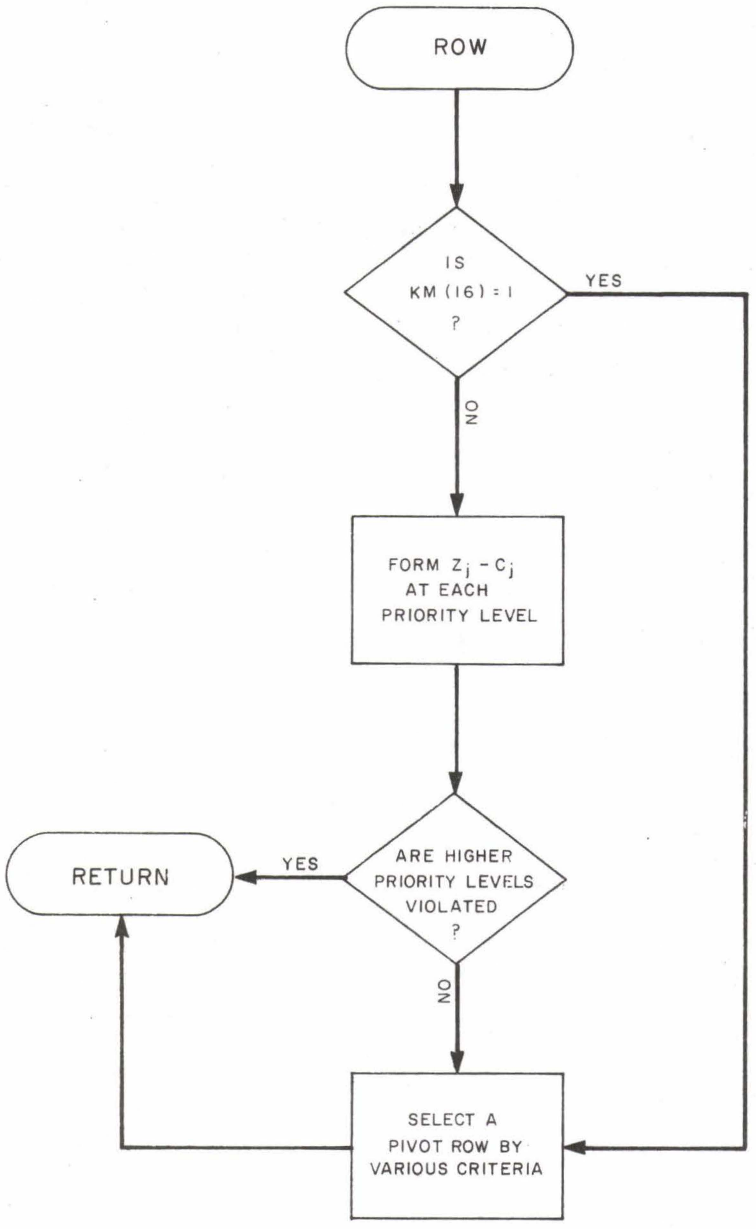
ROW
IS KM (16) = 1 ?
YES
NO
FORM $Z_j - C_j$ AT EACH PRIORITY LEVEL
ARE HIGHER PRIORITY LEVELS VIOLATED ?
YES
NO
RETURN
SELECT A PIVOT ROW BY VARIOUS CRITERIA

Fig. 14. Flow chart for SUBROUTINE ROW.

```
    DO 106 I=1,K2
    T(1)=0.0
    DO 102 J=1,M
    JJ=JH(J)
    IF(JG(JJ).NE.I) GO TO 102
    IF(ABS(Y(J)).LE.Z(1)) GO TO 102
    KB1=KB(JJ)
    IF(KN(JJ).EQ.POSD) GO TO 101
    T(1)=T(1)+Y(J)*WW(KB1)
    GO TO 102
101 T(1)=T(1)+Y(J)*DY(KB1)
102 CONTINUE
    IF(JG(KLM).NE.I) GO TO 105
    KB1=KB(KLM)
    IF(KN(KLM).EQ.NEGD) GO TO 103
    IF(KN(KLM).EQ.POSD) GO TO 104
103 T(1)=T(1)-WW(KB1)
    GO TO 105
104 T(1)=T(1)-DY(KB1)
105 IF(T(1).LE.-Z(8)) GO TO 117
106 CONTINUE
```

This block of code calculates the Z_j-C_j values at priority levels 1 to K2. T(1) is the Z_j-C_j value. If this value is ever negative, within a tolerance of Z(8), this means that the deviation from the goal at that priority level will be increased. When this happens, IR remains at zero (meaning that column KLM is rejected as a pivot column) and control is returned to CNTRL.

```
107 AA=0.0
    IAB=0
    DO 112 I=1,M
    IF(X(I).NE.0.0) GO TO 112
    YI=ABS(Y(I))
    IF(YI.LE.Z(1))GO TO 112
    IF(JH(I).EQ.0)GO TO 108
    IF(IAB.NE.0)GO TO 112
    IF(Y(I))112,112,109
108 IF(IAB)109,110,109
109 IF(YI-AA)112,112,111
110 IAB=1
111 AA=YI
    IR=I
112 CONTINUE
```

These instructions select a pivot row on the following criteria:

- -- the right-hand-side must be zero;
- -- the absolute value of the corresponding element of the pivot column must be greater than zero, *and the row must contain an artificial*;

-- and of the rows which meet the above two criteria, the row with the pivot column element of highest absolute value is chosen.

Or if no rows fit the above criteria, then the row which meets the following criteria is chosen.

-- the right-hand-side must be zero;

-- the value of the corresponding element of the pivot column must be greater than zero, and a non-artificial variable in solution in that row;

-- and of the rows which meet the above two criteria, the row with the pivot column element of highest positive value is chosen.

The variable IAB is zero until a row which meets the first set of criteria is found, and then the value is 1. Since the first set of criteria is top priority (driving artificials out of solution), as soon as IAB is 1, the second set of criteria is ignored.

The only reason that the second set of criteria exists is that it constitutes the minimum pivot ratio of zero (the right-hand-side values were zero). In this manner, the next set of instructions can be ignored (they find the minimum positive pivot ratio) if no artificials are in solution, since a row meeting the second set of criteria has been found.

```
 (i)          IF(IR.NE.0)GO TO 118
(ii)          Z(6)=0.0
              AA=1.0E+20
              DO 115 IT=1,M
              IF(Y(IT).LE.Z(1))GO TO 115
              IF(X(IT).LE.Z(2))GO TO 115
              XY=X(IT)/Y(IT)
              IF(XY-AA)114,113,115
          113 IF(JH(IT).NE.0)GO TO 115
          114 AA=XY
              Z(6)=XY
              IR=IT
          115 CONTINUE
```

(i) If a pivot row was found (IR will be non-zero) by the criteria of the last block of instructions, execution is sent to a RETURN.

(ii) If no pivot row has yet been found (IR is zero), then a search for a pivot row is made under the following criteria:

-- the element of the pivot column must be greater than zero;

-- the corresponding right-hand-side value must be greater than zero;

-- of the rows which meet the above two criteria, the one with the smallest pivot ratio is chosen (if there is a tie for the smallest ratio, and one of the tied rows contains an artificial, the row with the artificial is chosen).

```
                IF(KM(28).EQ.0)GO TO 118
```

KM(28) is 1 if a negative right-hand-side value exists. If this is not the case, then execution is sent to RETURN.

```
      (i)       BB=-Z(1)
                DO 116 I=1,M
                IF(X(I).GE.0.0))GO TO 116
                IF(Y(I).GE.BB)GO TO 116
                IF(Y(I)*AA.GT.X(I))GO TO 116
                BB=Y(I)
                IR=I
                Z(6)=X(I)/BB
            116 CONTINUE
                GO TO 118
     (ii)   117 KM(38)=1
            118 RETURN
```

If a negative right-hand-side value exists, this block of code is executed.

(i) For a pivot row to be chosen under this condition, the following criteria must be met:

-- the right-hand-side value must be less than zero;

-- the corresponding element of the pivot column must be negative;

-- the minimum pivot ratio from the positive rows times the element being considered in the pivot column must be less than or equal to the right-hand-side being considered. This simply insures that the considered pivot will not create more negatives than it eliminates.

-- Of those rows which meet the above three criteria, the row with the most negative pivot column element is chosen.

(ii) These instructions are executed when a previously minimized priority level is violated. When IR is returned to CNTRL as zero, KM(38) is checked for being 1. If this is the case, column KLM is rejected and a new pivot column is sought. If IR is zero and KM(38) is zero then an unbounded solution exists.

SUBROUTINE SEARCH (NAME, NUM) performs the task of comparing each command word with an integer value. This integer value will be used as the index in a computed GO TO in subroutine INPT which will direct execution to the proper instruction sequence. The flow chart for SUBROUTINE SEARCH is shown in Fig. 15.

Unique formal parameters:

NAME - This is the command word which was encountered elsewhere and is sent to SEARCH for interpretation.

NUM - This variable will contain the integer value associated with a command word and will be sent back to INPT.

```
      DATA  KNAME(1)/4HSTRT/, KNAME(2)/4HNAME/, KNAME(3)/4HRHS /
      DATA  KNAME(4)/4HMTRX/, KNAME(5)/4HEND /, KNAME(6)/4HSOLV/
      DATA  KNAME(7)/4HSTOP/, KNAME(8)/4HCHNG/, KNAME(9)/4HADBS/
      DATA KNAME(10)/4HARTR/,KNAME(11)/4HGO  /,KNAME(12)/4HSNGL/
      DATA KNAME(13)/4HCRSH/,KNAME(14)/4HINVT/,KNAME(15)/4HGOAB/
      DATA KNAME(16)/4HALTA/,KNAME(17)/4HALTB/,KNAME(18)/4HPARA/
```

Array KNAME is initialized with the command vocabulary. In this manner each command word is associated with a unique index value of KNAME. It is this index number which will be returned to INPT through NUM when a match is found between NAME and some element of KNAME.

```
      DO 101 K=1,18
      NUM=K
      IF(NAME.EQ.KNAME(K))GO TO 102
  101 CONTINUE
```

This DO loop compares NAME (the command word encountered) with each element of KNAME. When a match is found, the DO loop is exited to a RETURN statement, and NUM retains the index value of the element of KNAME in which the equivalent of NAME was found.

```
      WRITE(WRITEU,103)
  103 FORMAT(1H ,70HTHE JOB CONTROL CARD ENCOUNTERED IS EITHER NON-EXIST
     1ENT OR MISSPELLED./1H ,17HPLEASE TRY AGAIN.)
      NUM=100
  102 RETURN
```

If the DO loop goes to termination without finding a match for NAME, then the above diagnostic is printed. NUM is set to 100, which when returned to INPT, will send execution to read another command word.

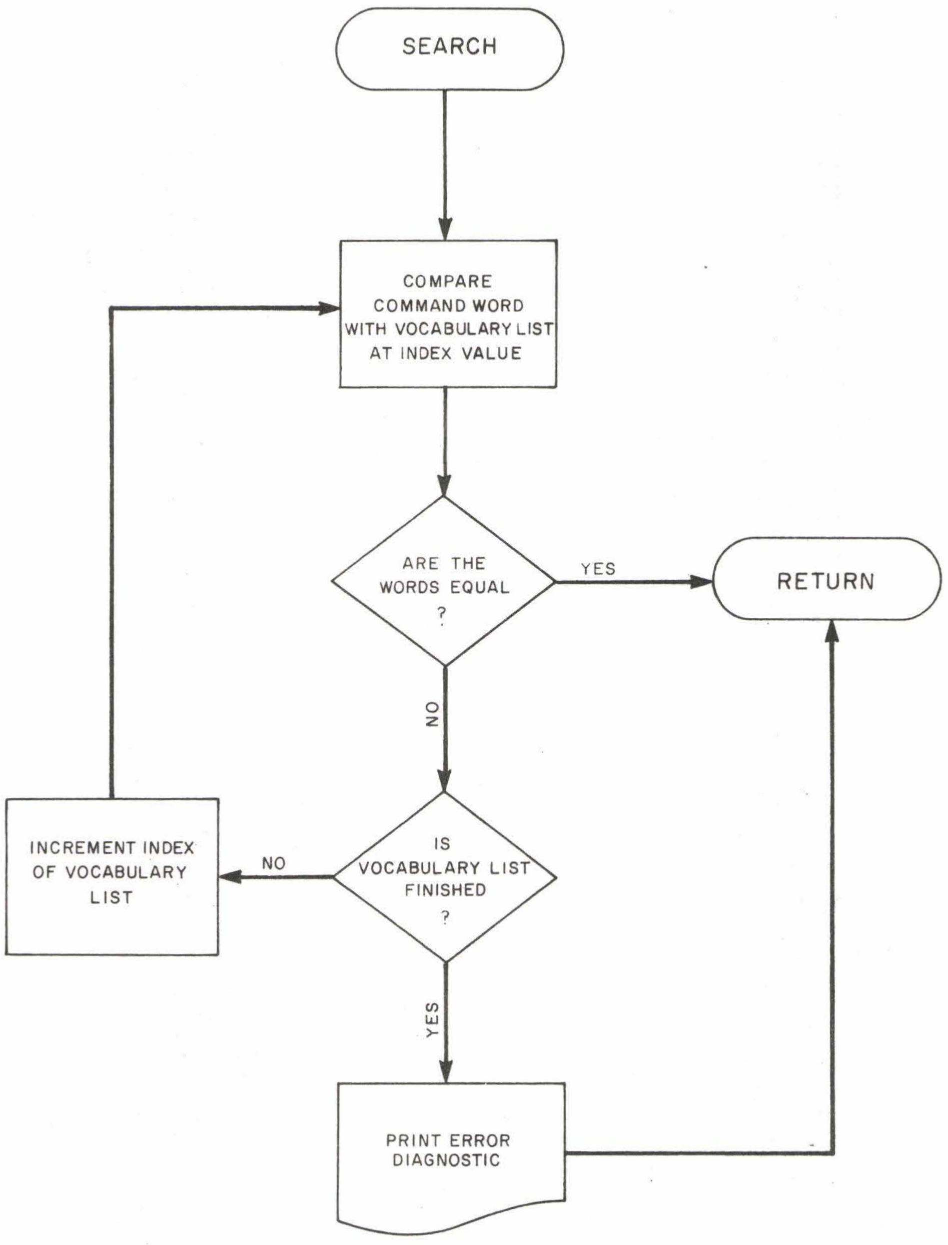
SEARCH
COMPARE COMMAND WORD WITH VOCABULARY LIST AT INDEX VALUE
ARE THE WORDS EQUAL ?
YES
RETURN
NO
IS VOCABULARY LIST FINISHED ?
NO
INCREMENT INDEX OF VOCABULARY LIST
YES
PRINT ERROR DIAGNOSTIC

Fig. 15. Flow chart for SUBROUTINE SEARCH.

SUBROUTINE XCK (K, COMMON PARAMETERS) serves two purposes:

(i) to identify the existence of any infeasibilities, and

(ii) to search for deviational variables in solution at the priority level being operated on.

The flow chart for SUBROUTINE XCK is shown in Fig. 16.

Unique formal parameters:

K - this variable is sent back to CNTRL as 1 if an infeasibility exists, or zero if none exists.

```
KM(28)=0
ITT=0
III=1
K=0
```

KM(28) is zero if no negative right-hand-side values exist and 1 otherwise. ITT is zero if no deviational variables are in solution at priority level KM(16) and 1 otherwise. III is 1 by default, and 2 if no infeasibilities exist and no deviational variables were found in solution after one pass of the solution array JH(I).

```
          101 DO 110 I=1,M
   (i)        GO TO (102,109),III
  (ii)    102 IF(ABS(X(I))-Z(2))103,103,104
          103 X(I)=0.0
              GO TO 105
 (iii)    104 IF(X(I))106,105,105
  (iv)    105 IF(JH(I))108,107,108
   (v)    106 KM(28)=1
              K=1
              GO TO 112
  (vi)    107 K=1
              GO TO 110
          108 IF(K.EQ.1) GO TO 110
 (vii)    109 JJ=JH(I)
              IF(JG(JJ).NE.KM(16)) GO TO 110
              ITT=1
          110 CONTINUE
```

(i) If III is 2, then one pass of the rows has already been made without finding any infeasibilities or deviational variables at priority level KM(16)-1; therefore, a search will be made for deviational variables in solution at KM(16) and infeasibilities will not be a concern.

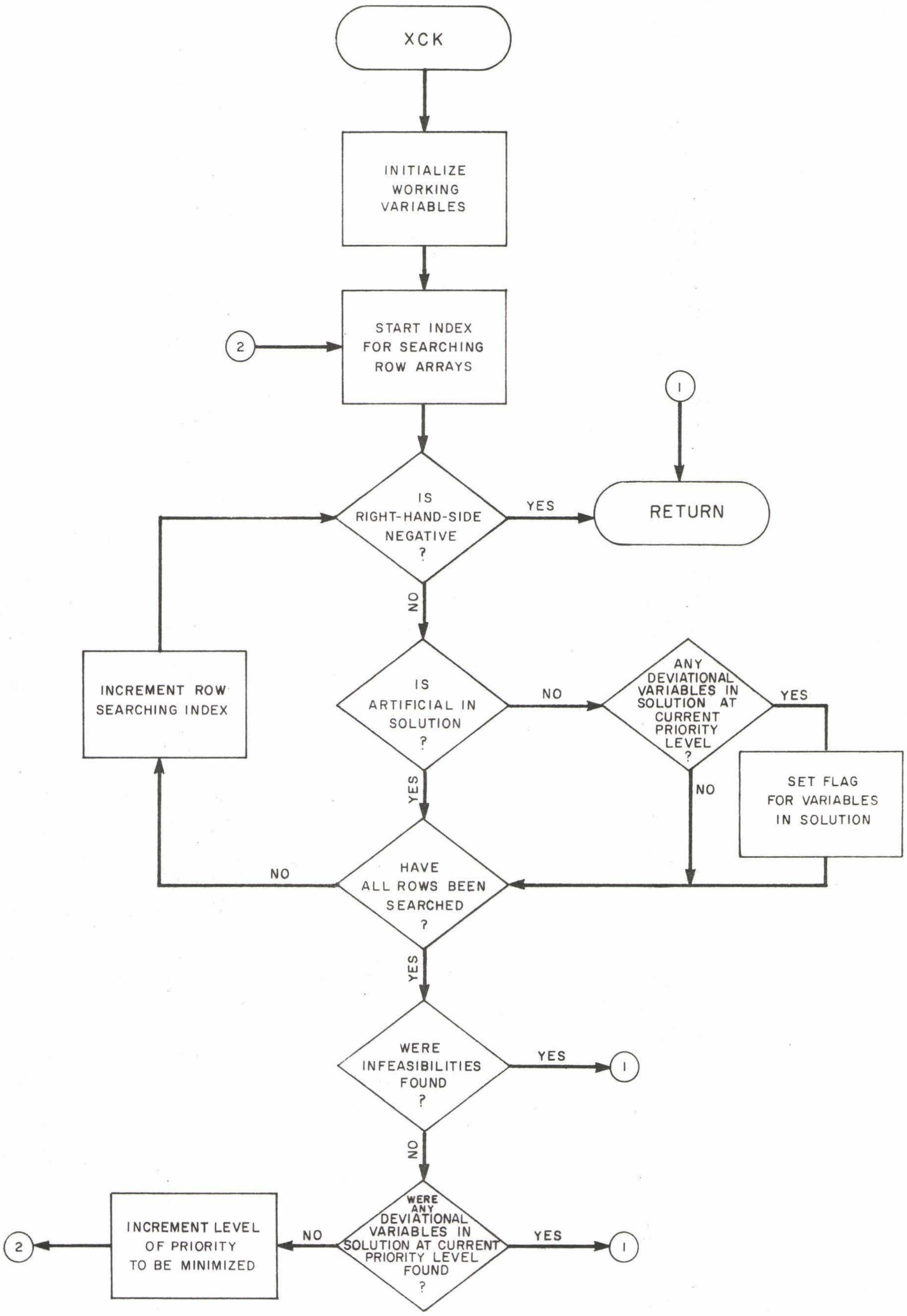
XCK
INITIALIZE WORKING VARIABLES
START INDEX FOR SEARCHING ROW ARRAYS
2
1
IS RIGHT-HAND-SIDE NEGATIVE ?
YES
RETURN
NO
IS ARTIFICIAL IN SOLUTION ?
NO
ANY DEVIATIONAL VARIABLES IN SOLUTION AT CURRENT PRIORITY LEVEL ?
YES
SET FLAG FOR VARIABLES IN SOLUTION
NO
YES
INCREMENT ROW SEARCHING INDEX
NO
HAVE ALL ROWS BEEN SEARCHED ?
YES
WERE INFEASIBILITIES FOUND ?
YES
1
NO
WERE ANY DEVIATIONAL VARIABLES IN SOLUTION AT CURRENT PRIORITY LEVEL FOUND ?
NO
YES
1
INCREMENT LEVEL OF PRIORITY TO BE MINIMIZED
2

Fig. 16. Flow chart for SUBROUTINE XCK.

(ii) Right-hand-side values sufficiently close to zero (within a tolerance of (Z(2)) are set to zero.

(iii) A check is made to see if a negative right-hand-side value exists.

(iv) JH(I) is checked for being zero, indicating that an artificial variable is in solution.

(v) These instructions are executed when a negative right-hand-side is found. As soon as this condition occurs, a RETURN is called.

(vi) Instructions are to be executed if an artificial in solution is found at (iv).

(vii) These are the instructions which check for deviational variables in solution at the priority level being operated on (KM(16)).

```
 (i)          IF(K.EQ.1) GO TO 112
(ii)          IF(ITT)112,111,112
          111 KM(16)=KM(16)+1
              IF(KM(16).GT.MB) GO TO 112
              III=2
              GO TO 101
          112 RETURN
              END
```

(i) After the DO loop is finished, if an infeasibility exists (K is 1) a RETURN is effected.

(ii) If no infeasibilities exist, then ITT is checked to see if any deviational variables were in solution at priority level KM(16). If so (ITT is 1) a RETURN is effected; if not, KM(16) is increased by 1 and checked to insure that it has not exceeded the number of priority levels (MB). If MB has not been exceeded, III is set to 2 so that no check will be made for infeasibilities and the DO loop is again initiated in search for deviational variables in solution at priority level KM(16).

DESCRIPTION OF PARAMETERS

Formal Parameters

Symbol	Dimension in words	Use
A	S	Non-zero technological coefficient entries.
B	M	Original right-hand-side values.
CAX	N	Current value of an objective function (routine BRO).
DY	M	Differential weight of positive deviational variables.
E	L	Pivot vector entries (eta file).
IA	S	Non-zero technological coefficient matrix entry row numbers.
IE	L	Row numbers for pivot element entries.
IKJ	N	Rows pivoted for columns.
IP	T	Column position of eta vector in implied inverse matrix.
IZIP	M	Constraint type designation.
JG	N	Column priority level.
JH	M	Column in row solution.
KB	N	Row number which generated column.
KG	M	Priority level of rows designated E, G, L or priority level of negative deviational variable of rows designated B.
KL	N	Number of first entry in a column.
KN	N	Column name.

KNT	M	Eligible pivot columns in routine BRO.
LABL	N+M -KM(14) -KM(26)	20 character description of rows and and computational columns.
LE	T	Column locator for eta file entries.
MG	M	Priority level for positive deviational variables in rows with B designation.
NR	M	Row names.
W	M	Values of current inverse matrix row.
WW	M	Differential weight for goal rows designated as G, L, E or for the negative deviational variables in rows designated as B.
X	M	Right-hand-side values for current solution vector.
Y	M	Matrix values for a pivot column.

where

L - maximum number of eta file entries.

M - maximum number of rows.

N - maximum number of columns including computational columns.

S - maximum number of non-zero entries in the technological coefficient matrix.

T - maximum number of pivots.

Variables in Common

Blank Common

D - This array is broken into the arrays listed as formal parameters. Altering the dimension of this array in the main program GOAL is the only change needed in accommodating different problem sizes.

Labeled Common

Symbol	Dimension in Words	Use
KBCD	30	Card image input storage.
*KM	50	Fixed point information.
M	1	Number of rows including objective functions.
MA	1	Number of technological coefficient entries.
MB	1	Number of goal priority levels.
ME	1	Number of pivots.
N	1	Number of columns including computational variables.
PRONAM	10	Problem description.
T	4	Temporary storage.
** Z	8	Floating point information.

* Fixed Point Information

KM(1)	Number of iterations.
KM(2)	Maximum number of non-zero matrix entries.
KM(3)	If 1 - problem requires artificial variables.
KM(4)	Inversion frequency.
KM(5)	Name of the pivot row.
KM(6)	Name of column in array Y.
KM(7)	Not used.
KM(8)	Number of artificial variables added.
KM(9)	Not used.
KM(10)	Number of eta file entries.
KM(11)	Maximum number of rows.

KM(12)	Maximum number of columns.
KM(13)	Eta file storage exceeded (1 if exceeded).
KM(14)	Number of surplus type variables.
KM(15)	Number of pivots performed.
KM(16)	Priority level being operated on.
KM(17)	Number of iterations since last inversion.
KM(18)	Characteristic of the logarithm.
KM(19)	Iteration number of last inversion.
KM(20)	Maximum number of pivots allowed.
KM(21)	Maximum number of eta file entries allowed.
KM(22)	Number of column pivot candidates.
KM(23)	Temporary storage.
KM(24)	Temporary storage.
KM(25)	4 alphanumeric blank characters.
KM(26)	Number of slack type variables.
KM(27)	Not used.
KM(28)	Negative right-hand-side flag (1 when negative).
KM(29)	Inversion frequency (nominally 300).
KM(30)	1 if infeasibilities exist.
KM(31)	Non-zero if intermediate results desired.
KM(32)	Number of basic artificials.
KM(33)	CRASH control (1 if crashing, zero if reinverting).
KM(34)	Nominally zero, 1 if initial solution needs to be created.
KM(35)	Control for type of output (1-long, 2-medium, 3-short, and 4-parametric run in progress).
KM(36)	Nominally zero, 1 if initial solution has been created.

KM(37)	Non-zero if echo of original data input desired.
KM(38)	1 if a higher priority level has been violated, zero if not.
KM(39)	Controls whether columns or rows are being flagged during advanced basis starts.

** Floating Point Information

Z(1)	Pivot tolerance.
Z(2)	Set right-hand-side to zero tolerance.
Z(3)	Objective function columns selection tolerance.
Z(4)	Relative zero tolerance in Y storage.
Z(5)	Set eta file entry to zero tolerance.
Z(6)	Current pivot ratio.
Z(7)	Antilogarithm of mantissa.
Z(8)	Upper priority level violation tolerance.

INPUT FORMATS

Data input and internal program control are accomplished through three types of control card sets: (i) type 0, (ii) type 1, and (iii) type N. Type 0 control cards are for specific actions to be taken by the program and have no data cards following them. Type 1 control cards have a single data card following them. Type N cards are followed by an arbitrary number of data cards. The data cards must have blanks in columns 1-4 and the data for the type N cards ends with an END card, punched in columns 1-3. All control cards must be punched left justified.

Type 0 Control Cards

CRSH	Appears before a GO command and after SNGL, or SNGL-INVT. This forces in as complete a basis as possible.
END	No action is caused. It terminates a set of data for type N control cards.
GO	Initiates execution of a problem from the initial basis solution only by simplexing.

Cols. 5-8	Blank for long output, MEDM for medium output, and SHRT for short output.

GOAB	Appears after ADBS and ARTR and solves the advanced basis start by inverting, crashing, and then simplexing. The same results could be obtained with the commands INVT, CRSH, and GO.
INVT	Inverts the eta file.
SNGL	Appears after the data sets RHS and MTRX and before any execution sequences other than SOLV or GOAB. This performs the creation of the additional computational variables used for initial basic solution.
SOLV	Causes the generation of a complete goal programming matrix, crashes the basis and solves. It is the equivalent of SNGL, CRSH and GO.

Cols. 5-8	Blank for long output, MEDM for medium output, and SHRT for short output.

STOP	Indicates the end of all control cards; stops the job.

Type 1 Control Cards

NAME	Appears anywhere before a SOLV card and is followed by a card containing an alphanumeric description of the problem in columns 1-40 and the date in columns 73-80. This card set is optional.
PARA	Appears after the RHS and MTRX data sets. This card performs sequential runs of a certain problem where the value of either a technological coefficient or a right-hand-side value is changed from lower limiting value to an upper limiting value by a given increment. The single data card which follows has this format:

Cols. 1-4	Blank
Cols. 5-8	Blank if run is being made on a right-hand-side value; name or symbol of column if run is being made on a technological coefficient.

Cols. 9-12	Name or symbol of row which corresponds to column name in columns 5-8 when run is performed on a technological coefficient; name or symbol of row when run is performed or a right-hand-side value. (This assumes that columns 5-8 are blank.)
Cols. 13-24	Value of lower limiting level of parameter.
Cols. 25-36	Value of upper limiting level of parameter.
Cols. 37-48	Value of the increments between the lower and upper level of the parameter.

STRT The first card of each problem to be run; initializes various constants and nominal values. The card to follow this one contains:

Cols. 1-5	Number of rows in the problem
Cols. 6-10	Number of decision variables (columns)
Cols. 11-15	Number of goal priority levels
Cols. 16-20	Number of MTRX data cards (non-zero matrix entries)
Cols. 21-25	Number of G type constraints
Cols. 26-30	Number of L type constraints
Cols. 31-35	Number of B type constraints
Cols. 36-40	Non-zero if intermediate results desired
Cols. 41-45	Non-zero if input data echo desired

Type N Control Cards

ADBS Appears after the RHS and MTRX cards or after any change card (ALTA or ALTB). This enters the columns that are to be in solution. The format of the data is as follows:

Cols. 1-4	Blank
Cols. 5-8	Number of a basic column
Cols. 9-12	Number of a basic column
Cols. 13-16	Number of a basic column
.	
.	
.	
Cols. 45-48	Number of a basic column
Cols. 49-80	Not used

As many data cards as necessary are used. This data set is ended by an END card as described in Type 0 Control Cards.

ALTA Appears anywhere after initial data entry and before a command sequence which will solve the problem. This card allows for changes in technological coefficients. The data cards which follow have this format:

Cols. 1-4	Blank
Cols. 5-8	Column name or symbol
Cols. 9-12	Row name or symbol
Cols. 13-24	Value of technological coefficient

This data set is ended with an END card as described in Type 0 Control Cards.

ALTB Appears anywhere after initial data entry and before a command sequence which will solve the problem. This card allows for changes in any of the data given on a data card following the RHS card except for the description of the row. A data card for each row with a change will appear after the ALTB card in the following format:

Cols. 1-8	Blank
Cols. 9-12	Name or symbol of row to be changed

All other columns will follow the format described as the data cards following the RHS card.

This data set is ended with an END card as described in the Type 0 Control Cards.

ARTR Appears immediately after the ADBS data set and before a solution sequence (INVT-CRSH-GO, INVT-GO, or GOAB). This indicates the name of the rows in which artificial variables are in solution. The data cards are as follows:

Cols. 1-4	Blank
Cols. 5-8	Number of a row with artificial
Cols. 9-12	Number of a row with artificial
Cols. 13-16	Number of a row with artificial
.	
.	
.	
Cols. 45-48	Number of a row with artificial
Cols. 49-80	Not used

As many data cards as necessary are used. This data set is ended by an END card as described in Type 0 Control Cards.

CHNG Appears anywhere after the start card and before the SOLV card. This card allows for changes in various tolerances and frequencies. The data cards which follow have this format:

Cols.	1-4	Blank
Cols.	5-8	TOLR if tolerance changes are to be made or FREQ if frequencies will be changed.
Col.	9	If columns 5-8 contain TOLR, column 9 will be

1 - change pivot tolerance
2 - change right-hand-side to zero tolerance
3 - change objective function tolerance
4 - change JMY tolerance
5 - change pivot entry tolerance
6 - not used
7 - not used
8 - change higher priority level violation tolerance.

If columns 5-8 contain FREQ then column 9 will be

1 - change inversion frequency
2 - change iteration cutoff frequency.

Cols.	10-25	Value to which the tolerance or frequency will be changed. Decimal is required.

TOLR must appear on each tolerance change card and FREQ on each frequency card. The two types of cards can be mixed at will after the CHNG card. This data set is ended with an END card as described in Type 0 Control Cards.

MTRX Appears after RHS control card and before SOLV. This card indicates the beginning of the data cards containing technological coefficient entries. The data cards which follow have this format:

Cols.	1-4	Blank
Cols.	5-8	Column number or symbol (alphanumeric field).
Cols.	9-12	Row number or symbol (alphanumeric field).
Cols.	13-24	Value of the technological entry with decimal punched.
Cols.	61-80	Description of the variable in that column which is optional.

All cards with the same columns must be together. This data set is ended with an END card as described in Type 0 Control cards.

RHS	Appears before the MTRX card, is followed by N data cards, with each card describing a row in the technological coefficient matrix in the following format:

Cols. 1-8	Blank
Cols. 9-12	Row number or symbol (alphanumeric information).
Cols. 13-24	Value of the right-hand-side of the row.
Col. 25	Type description of row relationship to the right-hand-side value. E - if equal G - if greater than or equal to L - if less than or equal to B - if both over- and underachievement of the goal level are acceptable. The B designation is used only in relation to goal constraints while E, G, or L may or may not be associated with a goal level.
Cols. 26-30	Blank if row has no association with a goal level. If associated with a goal level, the priority level at which deviations from the goal are minimized. This card field is related to: -- minimizing any deviation if column 25 is E. -- minimizing underachievement of the goal if column 25 is L. -- minimizing underachievement of the goal if column 25 is B.
Cols. 31-42	The differential weight associated with the priority level in columns 26-30. If columns 26-30 are blank, so will this field.
Cols. 43-47	Used only when column 25 is B and overachievement of the goal is to be minimized, or when column 25 is G and overachievement is to be minimized. These columns contain the priority level at which that minimization will take place.
Cols. 48-59	Differential weight associated with priority level in columns 43-47. If columns 43-47 are blank, so will this field.
Col. 60	Blank.
Cols. 61-80	20 alphanumeric character description of the row. This is optional.

Summary of Control Cards

Cols. 1-4	Type	Function	Format of Data
ADBS	N	Names of columns in basis	(A4, 11I4)
ALTA	N	Change technological coefficients	(3A4, F12.0)
ALTB	N	Change right-hand-side value	(3A4, F12.0, A1, 2(I5, F12.0))
ARTR	N	Names of rows which are artificial	(A4, 11I4)
CHNG	N	Change tolerances and/or frequencies	(2A4, I1, F16.8)
CRSH	0	Force in as complete a basis as possible	
END	0	No action - indicates end of a type N data set	
GO	0	Solve the problem from initial feasible solution only by simplexing	
GOAB	0	Composite of command INVT, CRSH, and GO for advanced basis start	
INVT	0	Invert	
MTRX	N	Matrix entries, one per card	(3A4, F12.0, 36X, 5A4)
NAME	1	Name of the problem	(10A4, 32X, 2A4)
PARA	1	Parametric run on matrix or right-hand-side values	(3A4, 3F12.0)
RHS	N	Right-hand-side entries	(3A4, F12.0, A1, 2(I5, F12.0), 1X, 5A4)

SNGL	0	Create all needed computational columns, and determine initial basic solution
SOLV	0	Composite for SNGL, CRSH, and GO
STOP	0	Stop execution of the program
STRT	1	Starts a new problem and reinitializes arrays and variables (9I5)

SPECIAL CONSIDERATIONS

Storage Technique

Non-zero matrix entries. Non-zero matrix entries are stored in a single array, A. The number of each matrix entry corresponds to the subscript of its array position. The array A is column oriented (i.e. the array entries for column one are together and come before the matrix entries of column two), but the ordering of the entries by rows within the column is not required.

A sister array to array A, called IA, also exists. Each array element of IA corresponds directly to its counterpart in A. IA contains the number of the row (of the matrix) in which the entry corresponding to the subscript value appears.

KL is the third array which completes the orientation of where a matrix entry is located in matrix position. KL is column oriented (one element of KL is allocated to each column) and contains the number of the first matrix entry of each column.

Example: A matrix in standard form is shown below.

$$\begin{bmatrix} 5 & 2 & 0 & 2 \\ 4 & 0 & 1 & 2 \\ 1 & 3 & 6 & 4 \end{bmatrix}$$

As stored for computer usage by arrays:

A(1)=5
A(2)=4
A(3)=1
A(4)=2
A(5)=3
A(6)=1
A(7)=6
A(8)=2
A(9)=2
A(10)=4

IA(1)=1
IA(2)=2
IA(3)=3
IA(4)=1
IA(5)=3
IA(6)=2
IA(7)=3
IA(8)=1
IA(9)=2
IA(10)=3

KL(1)=1
KL(2)=4
KL(3)=6
KL(4)=8

Eta vector (pivot vector) entries. Eta vector entries are stored very similarly to the matrix entries. E is the counterpart to A and entries in E are numbered as entries in the order of their creation. IE corresponds to IA and is the sister array to E. LE is the eta entry equivalent of KL with the exception that LE is pivot oriented (one element of LE is used for each eta vector).

The major difference in the eta vector entries is that each vector must be identified as to which column of the pivot matrix it represents. This is controlled by array IP (which is pivot oriented) which contains the number of the non-identity column for each pivot. The number of the row in the coefficient matrix which is pivoted into is also the number of the column in the pivot matrix to be non-identity.

Pricing Vector

In essence, a pricing vector is a single row taken from an inverse matrix. The multiplication of this vector by the original value of any column of the original matrix of coefficients will yield the current value of the matrix coefficient which intersects the number of the matrix column by the number of the inverse matrix row. This can be shown as:

$$(\underline{A}^{-1})^{i}\ \underline{A}_j = \bar{a}_{i,j}$$

where $(\underline{A}^{-1})^{i}$ is the ith row of the inverse matrix, $\underline{A}_j$ is the jth row of the matrix of coefficients and $a_{i,j}$ is the current value of the matrix entry at row i, column j.

When product form of the inverse is used in revised simplex, the inverse matrix is not explicitly expressed, but is generated as the product of simpler matrices (as this technique's name might imply). At each iteration only that portion of the inverse matrix which is required is generated in the following manner:

Original Matrix

$$\begin{bmatrix} 5 & 2 & 0 & 2 \\ 4 & 0 & 1 & 2 \\ 1 & 3 & 6 & 4 \end{bmatrix}$$

Pivot column 1 with row 3 yielding pivot (eta) vector 1 in product form:

IP(1)=3

$$\begin{bmatrix} -5 \\ -4 \\ 1 \end{bmatrix}$$

Now the current value of the matrix is

$$\begin{bmatrix} 0 & -13 & -30 & -18 \\ 0 & -12 & -23 & -14 \\ 1 & 3 & 6 & 4 \end{bmatrix}$$

Pivot column 4 with row 2 yields pivot vector 2 in product form:

IP(2)=2

$$\begin{bmatrix} -9/7 \\ -1/14 \\ 2/7 \end{bmatrix}$$

At this point the two pivot vectors generate the following inverse matrix:

$$\begin{bmatrix} 1 & -9/7 & 0 \\ 0 & -1/14 & 0 \\ 0 & 2/7 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -9/7 & 1/7 \\ 0 & -1/14 & 2/7 \\ 0 & 2/7 & -1/7 \end{bmatrix}$$

If it were desired to find row 1 of the above inverse matrix without the explicit matrix multiplication above, a more efficient procedure is:

IP(1)=3

$$\begin{bmatrix} 1 & -9/7 & 0 \end{bmatrix} \times \begin{bmatrix} -5 \\ -4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/7 \end{bmatrix}$$

yielding

$$\begin{bmatrix} 1 & -9/7 & 1/7 \end{bmatrix}$$

Note that the 1st row of the last pivot matrix is multiplied by the non-identity column of the next to the last pivot matrix, and the result (product) replaces the element of the row vector which corresponds to the column number of the non-identity vector. If more pivots had been performed, then the same process would occur with the newly established row vector being multiplied by the non-identity vector (this multiplication always goes through the pivot vectors in reverse order of their creation).

Another method for constructing only the first row of the inverse matrix is the use of a pricing vector. This is a row vector which is initialized to zero, except for a 1 in the element position corresponding to the row of the inverse matrix which is desired. After this initialization, multiplication back through the eta file proceeds as described above. For example, again row 1 of the inverse matrix is desired; therefore, the pricing vector is initialized as:

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Multiplying the pricing vector by the non-identity column of the last pivot matrix

IP(2)=2

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} -9/7 \\ -1/14 \\ 2/7 \end{bmatrix} = \begin{bmatrix} -9/7 \end{bmatrix}$$

yields an updated pricing vector of

$$\begin{bmatrix} 1 & -9/7 & 0 \end{bmatrix}$$

which is multiplied by the next to the last non-identity column pivot vector

IP(1)=3

$$\begin{bmatrix} 1 & -9/7 & 0 \end{bmatrix} \times \begin{bmatrix} -5 \\ -4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/7 \end{bmatrix},$$

yielding the current value of the 1st row of the inverse matrix

$$\begin{bmatrix} 1 & -9/7 & 1/7 \end{bmatrix}.$$

Implicit Goal Objective Functions

In GOAL, objective function coefficients are never entered into the matrix, but are created in their current value form at each iteration. This is accomplished by using a modified pricing vector concept coupled with the much used Z_j-C_j procedures of linear programming.

Modification of the pricing vector simply lies in the manner in which it is initialized. Two modifications of the pricing vector are:

(i) Initializing more than one element with the value of 1 results in a row vector (after multiplication through the eta file) which when multiplied by a column of the original matrix will yield the sum of the current values of the coefficients in the rows which correspond to the pricing vector positions initialized with the value of 1. This is done column by column.

(ii) By initializing the non-zero elements of the pricing vector with values other than 1, the current value of the coefficients will be proportional to the value at which the pricing vector was initialized (if initialized at 1.5 the coefficient will be 1.5 times its current value).

Combining these two modifications of the pricing vector with the Z_j-C_j concept, the objective function needs to be created only for the priority level which is being operated on. At each iteration a search is conducted through the row solution array. If a row is found in which a deviational variable at the priority level under consideration is in solution, the pricing vector element which corresponds to the solution row is initialized at the value of the differential weight of that deviational variable. This initialization occurs for each element of the pricing vector for which a deviational variable at the specified priority level is found.

A pricing vector, when initialized in this manner and multiplied by the eta file, will produce the Z_j when multiplied by a column of the original matrix. The C_j is only subtracted for those columns which are of the same priority level which produced the Z_j.

Tolerances

The product form of the inverse revised simplex technique is especially sensitive to infinite operations when applied on a digital computer. This is due to the additional multiplication and division demanded by the technique.

Without going into each tolerance and its specific characteristics, an attempt is made to caution users of some of the symptoms which indicate that tolerances need to be changed.

In goal programming the most common occurrence indicating a need for change is evidenced when certain goal levels are not attained, but resources needed for attainment are available. In this case Z(3), the objective function tolerance, needs to be a smaller number (i.e. the amount of contribution to the objective function for a column need not be so large before it is accepted as a pivot column).

Another common problem occurs when coefficients of great size (magnitude) differences exist in the matrix (i.e. 2,000,000.0 and .0005 in the same matrix). In this situation, making tolerances Z(1), Z(4) and Z(5) smaller might eradicate unreliable results. Under most circumstances, it is better to scale the matrix data.

When data does not require very low tolerances another problem might occur. Rounding residuals resulting from conversion of decimal to binary may generate incorrect coefficients and cause meaningless pivots. In this case Z(1), Z(4), and Z(5) should be made larger than their nominal values.

When examining the results from a goal programming solution, a good rule of thumb is to ask, "Are the results reasonable?" If not, the model should be checked for validity and then changes made in the tolerances.

In the past it has been found that a range exists within the tolerance configuration where the problem results remain constant. It is felt that when tolerance related problems are occurring for a given model, it is necessary to establish such a range (when a reasonable solution is found) in order to establish whether the results were model determined or tolerance determined.

EXAMPLE PROBLEM

The example problem deals with a land management decision. The land area in question consists of three homogeneous units: (i) bottomland, (ii) shrub, and (iii) shrub-lodgepole pine ecotone. Three management alternatives are available for each land type. These management alternatives will be the decision variables in the goal program with:

X_1 - acres of bottomland left as is

X_2 - acres of bottomland to drain

X_3 - acres of bottomland to spray

X_4 - acres of shrub land left as is

X_5 - acres where the shrub is mechanically removed

X_6 - acres where the shrub is mechanically removed and the land is seeded

X_7 - acres of ecotone land left as is

X_8 - number of 3-acre parcels of ecotone land to be developed as campgrounds

X_9 - acres of ecotone land to be developed as a wildlife habitat.

These variables will be represented in the matrix columns.

The implementation of the above management alternatives is limited by certain desired results (goals) and also by the resources available. These limitations (constraints) are defined by the rows of the matrix as shown below.

Row	Definition
1	the acres of bottomland
2	the acres of shrub land
3	the acres of shrub-lodgepole pine ecotone
4	tons of sediment produced by management alternatives (This is a goal constraint which strives to limit the production of sediment to below 67,875 tons at a priority level of 1.)
5	the available budget for implementation of alternatives
6	a profit goal (The constraint is at priority 5 and seeks to drive profit to as near 2,000,000 as possible.)
7	the desired number of cow-calf units (A goal constraint at priority 4 where at least 200 units are desired.)

8	the desired number of deer years (A goal constraint at priority 3 where at least 600 units are desired.)
9	the desired number of recreation user days (A goal constraint at priority 2 where at least 45,000 user days are desired.)

The relationships between the decision variables (X_1 through X_9) and the constraints (Rows 1 through 9) will be shown in the canonical description of the problem.

$$\text{Minimize:}\quad Z = P_1 d_4^+ + P_5 d_6^- + P_4 d_7^- + P_3 d_8^- + P_2 d_9^-$$

where P_i is the priority level for minimization and the subscript of d^+ or d^- indicates the row with which the deviation is associated, subject to the following constraints:1/

<u>Row</u>	
1	$X_1 + X_2 + X_3 = 570.$
2	$X_4 + X_5 + X_6 = 2000.$
3	$X_7 + 3.X_8 + X_9 = 480.$
4	$2.5X_1 + 6.5X_2 + 5.X_3 + 9.75X_4 + 10.X_5 + 9.25X_6 + 5.65X_7 + 25.35X_8 + 9.82X_9 \overset{<}{\underset{>}{-}} 67875.$
5	$125.X_2 + 3.5X_3 + 15.X_5 + 17.5X_6 + 2500.X_8 + 350.X_9 \leq 42250.$
6	$5.67X_1 - 118.47X_2 + 3.84X_3 + 3.67X_4 - 6.41X_5 - 6.43X_6 + 2.32X_7 - 2500.X_8 - 350.X_9 \leq 2000000.$
7	$1.05X_1 + 1.21X_2 + 1.36X_3 + .68X_4 + 1.59X_5 + 2.05X_6 + .43X_7 \overset{<}{\underset{>}{-}} 200.$
8	$.29X_1 + .19X_2 + .028X_3 + .47X_4 + .19X_5 + 1.67X_6 + .27X_7 + .56X_8 + .4X_9 \overset{<}{\underset{>}{-}} 600.$
9	$2700.X_8 \overset{<}{\underset{>}{-}} 45000.$

1/ The symbol $\overset{<}{\underset{>}{-}}$ denotes a goal instead of a fixed constraint.

The tableau is set up as follows:

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
1.	1.	1.	1.						
2.				1.	1.	1.			
3.							1.	3.	1.
4.	2.5	6.5	5.	9.75	10.	9.25	5.65	25.35	9.82
5.		125.0	3.5		15.	17.5		2500.	350.
6.	5.67	-118.47	3.84	3.67	-6.41	-6.43	2.32	-2500.	-350.
7.	1.05	1.21	1.36	.68	1.59	2.05	.43		
8.	.29	.19	.028	.47	.19	1.67	.27	.56	.4
9.								2700.	

Only the non-zero matrix entries are punched, and they are identified by the row and column identifier intersect. The data deck setup for entering the data into the program is shown in Table 2.

Once the data is entered, various operations can be performed. The most common case will be the SOLV command. The symbol (>) will indicate the input commands with the output generated following the command. The following results are obtained with this command:

```
> SOLV

CRASHING WITH    0 PIVOTS.
*BECAME INFEASIBLE, ITERATION    0
FEASIBLE ON ITERATION      7

THIS SECTION USED ONLY FOR ADVANCED BASIS STARTS.
THESE ARE THE NUMBERS OF THE COLUMNS IN SOLUTION.
      1   4   7   8  10  12  13  15  17

THESE ARE NUMBERS OF THE ROWS WITH ARTIFICIALS IN SOLUTION.
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$                                                                    $$
$$                                                                    $$
$$                   GOAL PROGRAMMING SOLUTION                        $$
$$                    - PROBLEM DESCRIPTION -                         $$
$$                                                                    $$
$$              EXAMPLE PROBLEM FOR TECHNICAL MANUAL                  $$
$$                                                                    $$
$$                           - DATE -                                 $$
$$                           01/15/76                                 $$
$$                                                                    $$
$$                                                                    $$
$$                                                                    $$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$



                                 CONSTRAINT SUMMARY

ROW  ROW RIGHT-          ROW              ROW       NEGATIVE              POSITIVE
     HAND-SIDE                                     DEVIATIONS            DEVIATIONS
NO.    VALUE         DESCRIPTION          TYPE  PRIORITY  WEIGHT     PRIORITY  WEIGHT


 1       570.00   ACRES BOTTOM LAND         E        0       .00        0        .00
 2      2000.00   ACRES SHRUB LAND          E        0       .00        0        .00
 3       480.00   ACRES SHRUB-LOD-ECO.      E        0       .00        0        .00
 4     67875.00   TONS OF SEDIMENT          B        0       .00        1       1.00
 5     42250.00   DOLLARS BUDGET            L        0       .00        0        .00
 6   2000000.00   DOLLARS PROFIT            L        5      1.00        0        .00
 7       200.00   COW-CALF UNITS            B        4      1.00        0        .00
 8       600.00   DEER YEARS                B        3      1.00        0        .00
 9     45000.00   RECREATION USER DAYS      B        2      1.00        0        .00

                            SUMMARY OF INPUT INFORMATION


NUMBER OF CONSTRAINT ROWS. . . . . . . . . . . . . . . . . . . . .   9
NUMBER OF NON-ZERO MATRIX ENTRIES. . . . . . . . . . . . . . . . .  60
NUMBER OF VARIABLES (COLUMNS). . . . . . . . . . . . . . . . . . .  19
NUMBER OF PRIORITIES . . . . . . . . . . . . . . . . . . . . . . .   5
NUMBER OF DECISION VARIABLES . . . . . . . . . . . . . . . . . . .   9
NUMBER OF POSITIVE DEVIATIONAL VARIABLES . . . . . . . . . . . . .   4
NUMBER OF NEGATIVE DEVIATIONAL VARIABLES . . . . . . . . . . . . .   6
NUMBER OF ARTIFICIAL VARIABLES . . . . . . . . . . . . . . . . . .   3
NUMBER OF ITERATIONS TO FIND THE SOLUTION. . . . . . . . . . . . .   9
```

```
                         OPTIMAL VALUE OF DECISION VARIABLES

     VARIABLE                     DESCRIPTION                  AMOUNT

       X1                   NO ACTION BOTTOM LAN               570.00
       X4                   NO ACTION SHRUB LAND              2000.00
       X7                   NO ACTION ECOTONE                  430.00
       X8                   CAMPGROUND ECOTONE                  16.67

                                GOAL ACHIEVEMENT

GOAL LEVEL     1 IS******************************COMPLETELY ACHIEVED.

GOAL LEVEL     2 IS******************************COMPLETELY ACHIEVED.

GOAL LEVEL     3 IS******************************COMPLETELY ACHIEVED.

GOAL LEVEL     4 IS******************************COMPLETELY ACHIEVED.

GOAL LEVEL     5 IS NOT ACHIEVED IN THE FOLLOWING CONSTRAINTS-
       *           6, DOLLARS PROFIT         ,
               IS UNDERACHIEVED BY   2030097.17 UNITS.
       *  SUMMARY-
                  GOAL 5 IS NOT ACHIEVED BY    2030097.17 WGTD UNITS.

                               GOAL SLACK ANALYSIS

     THIS SECTION ANALYZES GOAL CONSTRAINTS WITH -B- TYPE INEQUALITIES
     WHERE EITHER A NEGATIVE OR POSITIVE DEVIATION IS NOT GIVEN A PRIORITY
      LEVEL.  THE VALUE WILL THEN REFLECT THE AMOUNT BY WHICH THE EXACT GOAL
      WAS NOT ACHIEVED, EVEN THOUGH THE MINIMUM OR MAXIMUM GOAL LEVEL WAS
      ACHIEVED.

      ROW       GOAL                      EXACT           NEGATIVE       POSITIVE
    NUMBER   DESCRIPTION                GOAL LEVEL         SLACK          SLACK

       4    TONS OF SEDIMENT           67875.00         44098.00       0000.00

       7    COW-CALF UNITS               200.00          0000.00       1943.40

       8    DEER YEARS                   600.00          0000.00        630.73


                          RESOURCE UTILIZATION ANALYSIS

   ROW      RESOURCE          EXACT           RESOURCE          RESOURCE
 NUMBER   DESCRIPTION     RESOURCE LEVEL      NOT USED        OVER-PRODUCED

    5     DOLLARS BUDGET    42250.00           583.33            0000.00
```

If this problem had been solved previously, then it is possible to start from an advanced basis after the data has been entered. This can be done even if coefficients or right-hand-side values have been changed. The command sequence is as follows (note that the numbers which follow the ADBS command are taken from the previous run):

Table 2. Data for example problem (card image).

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

STRT
    9    9    5   50    0    2    4
NAME
EXAMPLE PROBLEM FOR TECHNICAL MANUAL                                    01/15/76
RHS
           1        570.E                                   ACRES BOTTOM LAND
           2       2000.E                                   ACRES SHRUB LAND
           3        480.E                                   ACRES SHRUB-LOD-ECO.
           4      67875.B                     1          1. TONS OF SEDIMENT
           5      42250.L                                   DOLLARS BUDGET
           6    2000000.L    5          1.                  DOLLARS PROFIT
           7        200.B    4          1.                  COW-CALF UNITS
           8        600.B    3          1.                  DEER YEARS
           9      45000.B    2          1.                  RECREATION USER DAYS
END
MTRX
      X1   1           1.                                    NO ACTION BOTTOM LAN
      X1   4          2.5
      X1   6         5.67
      X1   7         1.05
      X1   8          .29
      X2   1           1.                                    DRAIN BOTTOM LAND
      X2   4          6.5
      X2   5         125.
      X2   6      -118.47
      X2   7         1.21
      X2   8          .19
      X3   1           1.                                    SPRAY BOTTOM LAND
      X3   4           5.
      X3   5          3.5
      X3   6         3.84
      X3   7         1.36
      X3   8         .028
      X4   2           1.                                    NO ACTION SHRUB LAND
```

Table 2, continued

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      X4   4        9.75
      X4   6        3.67
      X4   7         .68
      X4   8         .47
      X5   2          1.                                    MECH REMOVAL SHRUB
      X5   4         10.
      X5   5         15.
      X5   6       -6.41
      X5   7        1.59
      X5   8         .19
      X6   2          1.                                    MECH REM+SEED SHRUB
      X6   4        9.25
      X6   5        17.5
      X6   6       -6.43
      X6   7        2.05
      X6   8        1.67
      X7   3          1.                                    NO ACTION ECOTONE
      X7   4        5.65
      X7   6        2.32
      X7   7         .43
      X7   8         .27
      X8   3          3.                                    CAMPGROUND ECOTONE
      X8   4       25.35
      X8   5       2500.
      X8   6      -2500.
      X8   8         .56
      X8   9       2700.
      X9   3          1.                                    WILDLIFE HAB ECOTONE
      X9   4        9.82
      X9   5        350.
      X9   6       -350.
      X9   8          .4
END
```

```
> ADBS
>         1   4   7   8  10  12  13  15  17
> END
> ARTR
> END
> GOAB
```

If any artificial variables were in solution, the numbers of the rows in which they were in solution would follow the ARTR command. The GOAB will generate the same results as the SOLV command, with the following exceptions:

```
ADVANCED BASIS START.    6 COLUMNS PIVOTED IN.
```

will appear before the problem heading, and

```
            NUMBER OF ITERATIONS TO FIND THE SOLUTION..     0
```

will be in the "SUMMARY OF INPUT INFORMATION".

At times it is desirable to change certain data input values. The following run shows the necessary command words and their data card for changing any data which was entered originally after the RHS command (the descriptions cannot be changed). Two items should be noted:

(i) The data cards following the ALTB command are in the same format as the RHS data, and

(ii) The SHRT which follows the SOLV command generates the short output where only the "OPTIMAL VALUE OF DECISION VARIABLES" is given.

In this run example, the priority levels of some of the goal constraint rows are shown (check the original value to see what the original priorities were).

```
>    ALTB
>              4     67875.B                   5          1.
>              6   2000000.L    2        1.
>              9     45000.B    1        1.
>    END
>    SOLVSHRT

  CRASHING WITH    0 PIVOTS.
*BECAME INFEASIBLE, ITERATION   0
FEASIBLE ON ITERATION      7

THIS SECTION USED ONLY FOR ADVANCED BASIS STARTS.
THESE ARE THE NUMBERS OF THE COLUMNS IN SOLUTION.
      1   4   7   8  10  12  13  15  17

THESE ARE NUMBERS OF THE ROWS WITH ARTIFICIALS IN SOLUTION.
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$                                                                    $$
$$                                                                    $$
$$                     GOAL PROGRAMMING SOLUTION                      $$
$$                                                                    $$
$$                       -PROBLEM DESCRIPTION-                        $$
$$               EXAMPLE PROBLEM FOR TECHNICAL MANUAL                 $$
$$                                                                    $$
$$                              -DATE-                                $$
$$                             01/15/76                               $$
$$                                                                    $$
$$                                                                    $$
$$                                                                    $$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


                 OPTIMAL VALUE OF DECISION VARIABLES

VARIABLE                    DESCRIPTION                   AMOUNT

   X1                  NO ACTION BOTTOM LAN               570.00
   X4                  NO ACTION SHRUB LAND              2000.00
   X7                  NO ACTION ECOTONE                  430.00
   X8                  CAMPGROUND ECOTONE                  16.67
```

The following example run has three characteristics of interest. The ALTB changes some priority levels (note how the results are affected from the original run). Since a SOLV was done previous to this run, an initial solution already exists; therefore, the sequence CRSH and GO causes the problem to be executed without recreating the initial solution. The last characteristic of interest is the MEDM which follows the GO command. This suppresses the "CONSTRAINT SUMMARY" and "SUMMARY OF INPUT INFORMATION" selections of the output.

The results are as follows:

```
>    ALTB
>              4     67875.B                      5          1.
>              6   2000000.L    1           1.
>    END
>    CRSH
     CRASHING WITH   0 PIVOTS.
>    GO  MEDM
```

```
*BECAME INFEASIBLE, ITERATION   0
FEASIBLE ON ITERATION      7

THIS SECTION USED ONLY FOR ADVANCED BASIS STARTS.
THESE ARE THE NUMBERS OF THE COLUMNS IN SOLUTION.
      1   4   7   10  12  13  15  17  18

THESE ARE NUMBERS OF THE ROWS WITH ARTIFICIALS IN SOLUTION.


     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
     $$                                                                    $$
     $$                                                                    $$
     $$                   GOAL PROGRAMMING SOLUTION                        $$
     $$                                                                    $$
     $$                    -PROBLEM DESCRIPTION-                           $$
     $$              EXAMPLE PROBLEM FOR TECHNICAL MANUAL                  $$
     $$                                                                    $$
     $$                           -DATE-                                   $$
     $$                          01/15/76                                  $$
     $$                                                                    $$
     $$                                                                    $$
     $$                                                                    $$
     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


                     OPTIMAL VALUE OF DECISION VARIABLES

     VARIABLE                  DESCRIPTION                      AMOUNT

       X1                  NO ACTION BOTTOM LAN                 570.00
       X4                  NO ACTION SHRUB LAND                2000.00
       X7                  NO ACTION ECOTONE                    480.00


                             GOAL ACHIEVEMENT

GOAL LEVEL     1 IS NOT ACHIEVED IN THE FOLLOWING CONSTRAINTS-
       *          6, DOLLARS PROFIT         ,
               IS UNDERACHIEVED BY   1988314.50 UNITS.
       * SUMMARY-
            GOAL      1 IS NOT ACHIEVED BY   1988314.50 WGTD UNITS.
GOAL LEVEL     2 IS NOT ACHIEVED IN THE FOLLOWING CONSTRAINTS-
       *          9, RECREATION USER DAYS,
               IS UNDERACHIEVED BY      45000.00 UNITS.
       * SUMMARY-
            GOAL      2 IS NOT ACHIEVED BY      45000.00 WGTD UNITS.

GOAL LEVEL     3 IS******************************COMPLETELY ACHIEVED.

GOAL LEVEL     4 IS******************************COMPLETELY ACHIEVED.

GOAL LEVEL     5 IS******************************COMPLETELY ACHIEVED.
```

```
                          GOAL SLACK ANALYSIS
    THIS SECTION ANALYZES GOAL CONSTRAINTS WITH -B- TYPE INEQUALITIES
    WHERE EITHER A NEGATIVE OR POSITIVE DEVIATION IS NOT GIVEN A PRIORITY
    LEVEL.  THE VALUE WILL THEN REFLECT THE AMOUNT BY WHICH THE EXACT GOAL
    WAS NOT ACHIEVED, EVEN THOUGH THE MINIMUM OR MAXIMUM GOAL LEVEL WAS
    ACHIEVED.


ROW           GOAL              EXACT             NEGATIVE        POSITIVE
NUMBER      DESCRIPTION       GOAL LEVEL            SLACK           SLACK

  4         TONS OF SEDIMENT   67875.00           44238.00        0000.00
  7         COW-CALF UNITS       200.00            0000.00        1964.90
  8         DEER YEARS           600.00            0000.00         634.90


                      RESOURCE UTILIZATION ANALYSIS
ROW          RESOURCE              EXACT           RESOURCE        RESOURCE
NUMBER      DESCRIPTION       RESOURCE LEVEL       NOT-USED     OVER-PRODUCED

  5         DOLLARS BUDGET        42250.00         42250.00        0000.00
```

The last example run deals with parametric runs. This feature is designed to exhibit sensitivity of the basis to changes in technological coefficients or right-hand-side values. A parametric run can only be made on one value at a time. The example shows a parametric solution on the right-hand-side of row 9 from the lower bound of 20000 to the upper bound of 30000, with a solution given at each increment of 5000 between the two bounds (parameters). The results are as follows:

```
>     PARA
>               9      20000.      30000.      5000.


    $$$$$$$$$$$$$$$$$$$$$$$$$$$$PARAMETRIC RUN$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
    $$                                 ON                                   $$
    $$                   RIGHT-HAND-SIDE VALUE OF ROW      9                $$
    $$                           AT A VALUE OF                              $$
    $$                            200000.000                                $$
    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
ADVANCED BASIS START.    6 COLUMNS PIVOTED IN.

THIS SECTION USED ONLY FOR ADVANCED BASIS STARTS.
THESE ARE THE NUMBERS OF THE COLUMNS IN SOLUTION.
     1   4   7   8  10  12  13  15  17

THESE ARE NUMBERS OF THE ROWS WITH ARTIFICIALS IN SOLUTION.

     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
     $$                                                                     $$
     $$                                                                     $$
     $$                     GOAL PROGRAMMING SOLUTION                       $$
     $$                                                                     $$
     $$                      -PROBLEM DESCRIPTION-                          $$
     $$                EXAMPLE PROBLEM FOR TECHNICAL MANUAL                 $$
     $$                                                                     $$
     $$                              -DATE-                                 $$
     $$                             01/15/76                                $$
     $$                                                                     $$
     $$                                                                     $$
     $$                                                                     $$
     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


                              CONSTRAINT SUMMARY

ROW   ROW RIGHT-        ROW              ROW      NEGATIVE          POSITIVE
NO.   HAND-SIDE      DESCRIPTION         TYPE    DEVIATIONS        DEVIATIONS
        VALUE                                 PRIORITY  WEIGHT  PRIORITY  WEIGHT

 1       570.00   ACRES BOTTOM LAND       E        0        .00      0        .00
 2      2000.00   ACRES SHRUB LAND        E        0        .00      0        .00
 3       480.00   ACRES SHRUB-LOD-ECO.    E        0        .00      0        .00
 4     67875.00   TONS OF SEDIMENT        B        0        .00      1       1.00
 5     42250.00   DOLLARS BUDGET          L        0        .00      0        .00
 6   2000000.00   DOLLARS PROFIT          L        5       1.00      0        .00
 7       200.00   COW-CALF UNITS          B        4       1.00      0        .00
 8       600.00   DEER YEARS              B        3       1.00      0        .00
 9     20000.00   RECREATION USER DAYS    B        2       1.00      0        .00

                          SUMMARY OF INPUT INFORMATION

           NUMBER OF CONSTRAINT ROWS. . . . . . . . . . . . . . . . .  9
           NUMBER OF NON-ZERO MATRIX ENTRIES. . . . . . . . . . . . . 60
           NUMBER OF VARIABLES (COLUMNS). . . . . . . . . . . . . . . 19
           NUMBER OF PRIORITIES . . . . . . . . . . . . . . . . . . .  5
           NUMBER OF DECISION VARIABLES . . . . . . . . . . . . . . .  9
           NUMBER OF POSITIVE DEVIATIONAL VARIABLES . . . . . . . . .  4
           NUMBER OF NEGATIVE DEVIATIONAL VARIABLES . . . . . . . . .  6
           NUMBER OF ARTIFICIAL VARIABLES . . . . . . . . . . . . . .  3
           NUMBER OF ITERATIONS TO FIND THE SOLUTION. . . . . . . . .  0
```

```
                    OPTIMAL VALUE OF DECISION VARIABLES

    VARIABLE                DESCRIPTION                   AMOUNT

       X1              NO ACTION BOTTOM LAN                570.00
       X4              NO ACTION SHRUB LAND               2000.00
       X7              NO ACTION ECOTONE                   457.78
       X8              CAMPGROUND ECOTONE                    7.41


                              GOAL ACHIEVEMENT

GOAL LEVEL      1 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      2 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      3 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      4 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      5 IS NOT ACHIEVED IN THE FOLLOWING CONSTRAINTS-
       *          6, DOLLARS PROFIT      ,
                IS UNDERACHIEVED BY   2006884.57 UNITS.
       * SUMMARY-
           GOAL     5 IS NOT ACHIEVED BY    2006884.57 WGTD UNITS.


                            GOAL SLACK ANALYSIS

    THIS SECTION ANALYZES GOAL CONSTRAINTS WITH -B- TYPE INEQUALITIES
    WHERE EITHER A NEGATIVE OR POSITIVE DEVIATION IS NOT GIVEN A PRIORITY
    LEVEL.  THE VALUE WILL THEN REFLECT THE AMOUNT BY WHICH THE EXACT GOAL
    WAS NOT ACHIEVED, EVEN THOUGH THE MINIMUM OR MAXIMUM GOAL LEVEL WAS
    ACHIEVED.


    ROW        GOAL              EXACT          NEGATIVE       POSITIVE
    NO.     DESCRIPTION        GOAL LEVEL         SLACK          SLACK

     4      TONS OF SEDIMENT  67875.00         44175.78       0000.00
     7      COW-CALF UNITS      200.00          0000.00       1955.34
     8      DEER YEARS          600.00          0000.00        633.05


                      RESOURCE UTILIZATION ANALYSIS

ROW        RESOURCE            EXACT          RESOURCE          RESOURCE
NO.       DESCRIPTION     RESOURCE LEVEL      NOT USED        OVER-PRODUCED

 5      DOLLARS BUDGET       42250.00         23731.48          0000.00
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$PARAMETRIC RUN$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$                                 ON                                 $$
$$                    RIGHT-HAND-SIDE VALUE OF ROW      9             $$
$$                           AT A VALUE OF                            $$
$$                             25000.000                              $$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

ADVANCED BASIS START.    6 COLUMNS PIVOTED IN.


                     OPTIMAL VALUE OF DECISION VARIABLES

        VARIABLE            DESCRIPTION                      AMOUNT

          X1             NO ACTION BOTTOM LAN                 570.00
          X4             NO ACTION SHRUB LAND                2000.00
          X7             NO ACTION ECOTONE                    452.22
          X8             CAMPGROUND ECOTONE                     9.26


                              GOAL ACHIEVEMENT

GOAL LEVEL      1 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      2 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      3 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      4 IS****************************COMPLETELY ACHIEVED.

GOAL LEVEL      5 IS NOT ACHIEVED IN THE FOLLOWING CONSTRAINTS-
     *               6, DOLLARS PROFIT       ,
                IS UNDERACHIEVED BY    2011527.09 UNITS.
     * SUMMARY-
          GOAL    5 IS NOT ACHIEVED BY    2011527.09 WGTD UNITS.


                             GOAL SLACK ANALYSIS

   THIS SECTION ANALYZES GOAL CONSTRAINTS WITH -B- TYPE INEQUALITIES
   WHERE EITHER A NEGATIVE OR POSITIVE DEVIATION IS NOT GIVEN A PRIORITY
   LEVEL.  THE VALUE WILL THEN REFLECT THE AMOUNT BY WHICH THE EXACT GOAL
   WAS NOT ACHIEVED, EVEN THOUGH THE MINIMUM OR MAXIMUM GOAL LEVEL WAS
   ACHIEVED.

   ROW        GOAL                EXACT          NEGATIVE          POSITIVE
   NO.    DESCRIPTION           GOAL LEVEL         SLACK             SLACK

    4   TONS OF SEDIMENT        67875.00         44160.22          0000.00

    7   COW-CALF UNITS            200.00          0000.00          1952.96

    8   DEER YEARS                600.00          0000.00           632.59
```

```
                          RESOURCE UTILIZATION ANALYSIS

ROW       RESOURCE              EXACT          RESOURCE          RESOURCE
NO.     DESCRIPTION        RESOURCE LEVEL     NOT-USED        OVER-PRODUCED

 5     DOLLARS BUDGET         42250.00        19101.85           0000.00


     $$$$$$$$$$$$$$$$$$$$$$$$$$$$PARAMETRIC RUN$$$$$$$$$$$$$$$$$$$$$$$$$$$$
     $$                              ON                                  $$
     $$                RIGHT-HAND-SIDE VALUE OF ROW      9               $$
     $$                           AT A VALUE OF                          $$
     $$                             30000.000                            $$
     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

ADVANCED BASIS START.    6 COLUMNS PIVOTED IN.


                      OPTIMAL VALUE OF DECISION VARIABLES

          VARIABLE              DESCRIPTION                 AMOUNT

             X1           NO ACTION BOTTOM LAN               570.00
             X4           NO ACTION SHRUB LAND              2000.00
             X7           NO ACTION ECOTONE                  446.67
             X8           CAMPGROUND ECOTONE                  11.11


                              GOAL ACHIEVEMENT

GOAL LEVEL     1 IS*****************************COMPLETELY ACHIEVED.

GOAL LEVEL     2 IS*****************************COMPLETELY ACHIEVED.

GOAL LEVEL     3 IS*****************************COMPLETELY ACHIEVED.

GOAL LEVEL     4 IS*****************************COMPLETELY ACHIEVED.

GOAL LEVEL     5 IS NOT ACHIEVED IN THE FOLLOWING CONSTRAINTS-
        *         6, DOLLARS PROFIT       ,
               IS UNDERACHIEVED BY   2016169.61 UNITS.
        * SUMMARY-
           GOAL      5 IS NOT ACHIEVED BY   2016169.61 WGTD UNITS.


                            GOAL SLACK ANALYSIS

    THIS SECTION ANALYZES GOAL CONSTRAINTS WITH -B- TYPE INEQUALITIES
    WHERE EITHER A NEGATIVE OR POSITIVE DEVIATION IS NOT GIVEN A PRIORITY
    LEVEL.  THE VALUE WILL THEN REFLECT THE AMOUNT BY WHICH THE EXACT GOAL
    WAS NOT ACHIEVED, EVEN THOUGH THE MINIMUM OR MAXIMUM GOAL LEVEL WAS
    ACHIEVED.
```

```
ROW        GOAL                 EXACT        NEGATIVE        POSITIVE
NO.     DESCRIPTION          GOAL LEVEL       SLACK           SLACK

 4     TONS OF SEDIMENT       67875.00       44144.67         0000.00
 7     COW-CALF UNITS           200.00        0000.00         1950.57
 8     DEER YEARS               600.00        0000.00          632.12

                    RESOURCE UTILIZATION ANALYSIS

ROW      RESOURCE          EXACT            RESOURCE         RESOURCE
NO.     DESCRIPTION     RESOURCE LEVEL      NOT USED       OVER-PRODUCED

 5     DOLLARS BUDGET     42250.00          14472.22         0000.00

> STOP
```

Note that in the above example the solution variables were not sensitive to the right-hand-side value of row 9 between the values of 20,000 and 30,000. The value of some of the variables changed, but the solution variables were consistent.

The examples given in this section do not show all of the different options available to the user once the input data has been read by GOAL. For more details reference to the Input Format section of this manual or the General User's Manual - GOAL should be made. These two references allude to the use of individual command words and also sequences of command words which are needed to accomplish certain tasks.

GLOSSARY

A Matrix - matrix of technological coefficients.

Advanced Basis - a basis which has been optimized beyond the point of an initial solution. This basis is identified by solution columns, and rows with artificial variables in solution.

Artificial Variable - a variable required by constraint rows of the "greater than or equal to" type or the "equal to" type when the initial solution is formed.

Basis - those variables (columns) which are in solution.

Coefficient - see technological coefficient.

Computational Variable - a variable, other than a decision variable, which is added to the goal programming problem to aid in its solution; these include slacks, surpluses, artificials, and deviational variables.

Crashing - see feasible crashing.

Decision Variable - a variable whose optimal value is to be found by the goal programming procedure.

Deviational Variable - a variable representing over- or underachievement of a goal level.

Differential Weight - a value assigned to a priority factor for minimizing a deviational variable which has been assigned a priority level.

Eta File - constituted of all eta (pivot) vectors together.

Eta Vector - that column vector which contains the non-identity column of a pivot matrix.

Feasible Crashing - process which attempts to remove artificial variables from the basis in the fastest possible manner.

Goal Level - right-hand-side value associated with a goal constraint.

Infeasibility - situation where an artificial variable is in solution or a negative right-hand-side exists.

Initial Solution - that matrix configuration in which each row has a column in solution and the coefficient of the intersect with the column in solution is a positive one.

Matrix Entry - see technological coefficient.

Parametric Run - execution of a problem where a solution is generated at given incremental values between two specified bounds of a technological coefficient or right-hand-side value.

Pivot Column - that column of the matrix which is selected for entry into solution.

Pivot Element - that coefficient which lies at the intersect of the pivot row and column.

Pivot Row - that row which is most constraining when a column is being considered for entry into solution.

Pivot Vector - see eta vector.

Priority Level - an integer assigned to a deviational variable, indicating the order in which its minimization is to take place in the goal programming procedure. (E.g., a deviational variable assigned priority level 2 will be minimized before one at priority level 3, but never at the expense of one at priority level 1.)

Reinversion - that process where the inverse matrix, as expressed in product form eta vectors, is reformed so that the inverse matrix representation will be expressed in the fewest eta vectors possible.

Right-hand-side - the limiting value at which a row is constrained.

Technological coefficient - that numerical value which describes the relationship between the variable columns and constraint rows of the A matrix.